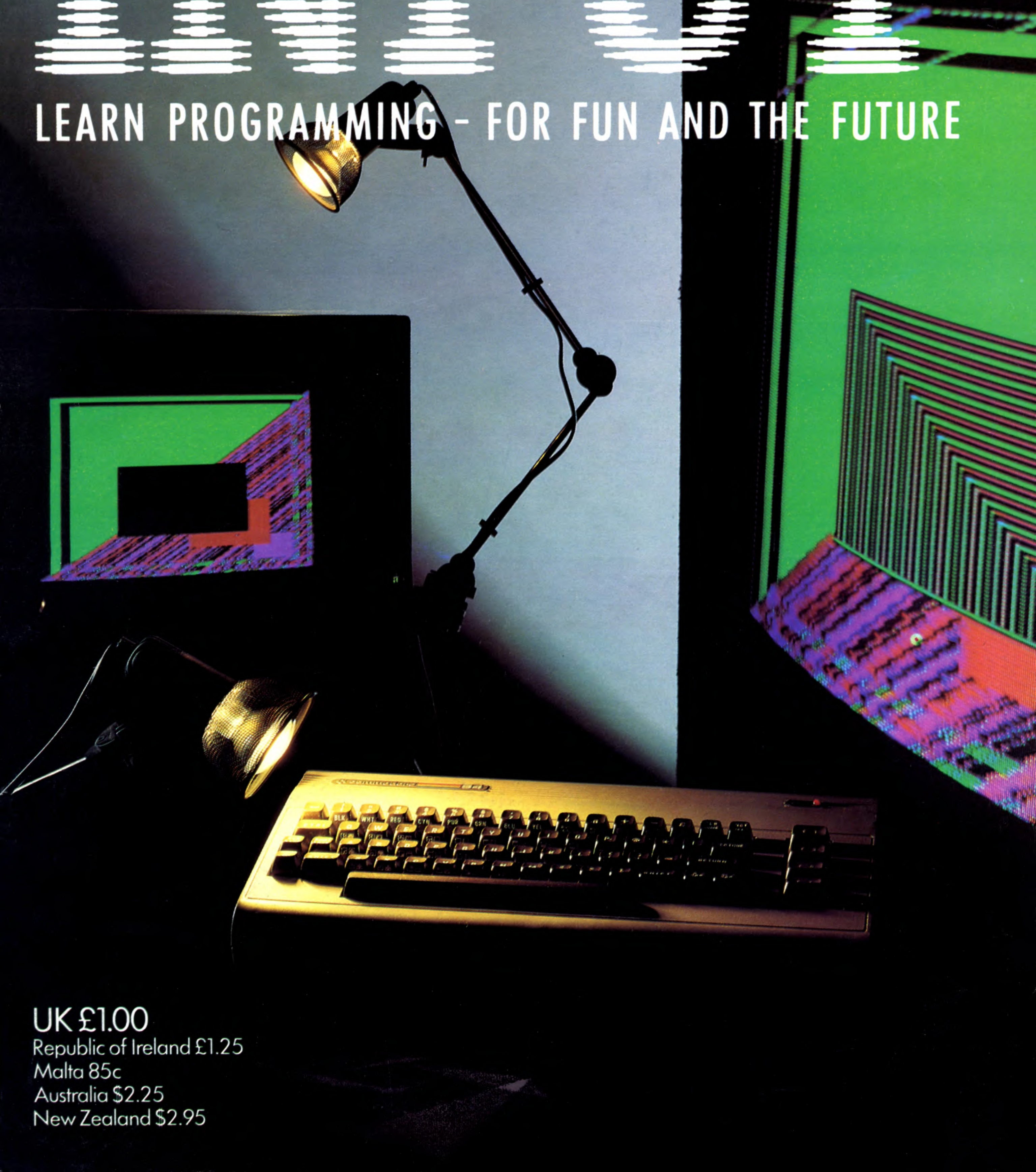
A MARSHALL CAVENDISH **28** COMPUTER COURSE IN WEEKLY PARTS

INPUT

LEARN PROGRAMMING - FOR FUN AND THE FUTURE



UK £1.00
Republic of Ireland £1.25
Malta 85c
Australia $2.25
New Zealand $2.95

INPUT

Vol. 3 **No 28**

INDEX

The last part of INPUT, Part 52, will contain a complete, cross-referenced index. For easy access to your growing collection, a cumulative index to the contents of each issue is contained on the inside back cover.

PICTURE CREDITS
Front cover, Dave King. Pages 857, 858, Ian Stephen. Pages 858, 859, Berry Fallon Design. Pages 860, 861, 862, Digital Arts. Pages 865, 866, 870, Johanne Ryder. Pages 872, 874, Dave King. Pages 878, 881, 882, Kevin O'Brien. Pages 884, 886, Graeme Harris.

Published by Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA, England. Printed by Artisan Presss, Leicester and Howard Hunt Litho, London.



There are four binders each holding 13 issues.

HOW TO ORDER YOUR BINDERS

UK and Republic of Ireland:
Send £4.95 (inc p & p) (IR£5.95) for each binder to the address below:
Marshall Cavendish Services Ltd, Department 980, Newtown Road, Hove, Sussex BN3 7DN

Australia: See inserts for details, or write to INPUT, Times Consultants, PO Box 213, Alexandria, NSW 2015

New Zealand: See inserts for details, or write to INPUT, Gordon and Gotch (NZ) Ltd, PO Box 1595, Wellington

Malta: Binders are available from local newsagents.

BACK NUMBERS

Back numbers are supplied at the regular cover price (subject to availability).

UK and Republic of Ireland:
INPUT, Dept AN, Marshall Cavendish Services, Newtown Road, Hove BN3 7DN

Australia, New Zealand and Malta:
Back numbers are available through your local newsagent.

COPIES BY POST

Our Subscription Department can supply copies to any UK address regularly at £1.00 each. For example the cost of 26 issues is £26.00; for any other quantity simply multiply the number of issues required by £1.00. Send your order, with payment to:

Subscription Department, Marshall Cavendish Services Ltd, Newtown Road, Hove, Sussex BN3 7DN

Please state the title of the publication and the part from which you wish to start.

HOW TO PAY: Readers in UK and Republic of Ireland: All cheques or postal orders for binders, back numbers and copies by post should be made payable to:
Marshall Cavendish Partworks Ltd.

QUERIES: When writing in, please give the make and model of your computer, as well as the Part No., page and line where the program is rejected or where it does not work. We can only answer specific queries – and please do not telephone. Send your queries to INPUT Queries, Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA.

INPUT IS SPECIALLY DESIGNED FOR:

The SINCLAIR ZX SPECTRUM (16K, 48K, 128 and +), COMMODORE 64 and 128, ACORN ELECTRON, BBC B and B+, and the DRAGON 32 and 64.

In addition, many of the programs and explanations are also suitable for the SINCLAIR ZX81, COMMODORE VIC 20, and TANDY COLOUR COMPUTER in 32K with extended BASIC. Programs and text which are specifically for particular machines are indicated by the following symbols:

SPECTRUM 16K, 48K, 128, and +
COMMODORE 64 and 128
ACORN ELECTRON, BBC B and B+
DRAGON 32 and 64
ZX81
VIC 20
TANDY TRS80 COLOUR COMPUTER

CONES, CURVES AND CUTS

- SLICING THE CONE
- DRAWING A CIRCLE, ELLIPSE, PARABOLA AND HYPERBOLA
- ROTATING THE CURVES
- PRACTICAL APPLICATIONS

The simple cone is one of the most fascinating mathematical shapes, producing a whole family of important curves. Here are some programs to explore its attributes

Curves have fascinated mathematicians from the earliest days, and the simpler and more elegant the curves were, the more important they seemed. The early Greek mathematicians were very keen to keep maths as simple as possible, so when it was discovered that an entire family of curves—known as conic sections—could be obtained simply by slicing through a cone, it seemed obvious that cones must have some special significance. Depending on how the cone is sliced, you can obtain a circle, an ellipse a parabola or a hyperbola.

In fact the beauty of these curves is that they are not mere mathematical abstractions, but crop up time and again in everyday life, and provide an accurate description of real physical phenomena.

There are, of course, other simple curves that are found in nature that are not sections of a cone. The shape of a rope or chain hanging between two points is one. It is called a catenary, and even though it looks rather like a parabola it is subtly different, and is described by quite different equations. But the conic sections are important as they are often related to the way things move and so are needed for any realistic program.

Some of the curves are also useful as solid three-dimensional objects. The slices through a cone are obviously two-dimensional, but they can be rotated round their axis to form a three-dimensional shape. The circle becomes a sphere, with any number of uses, and the parabola becomes a paraboloid, used in things as diverse as car headlamps, telescope mirrors, solar furnaces and many others.

This article is in two parts. The first part describes each curve and how to draw it on the screen, while the second part shows how to use the curves in simulations such as the path of a bucket (or person) attached to a slipping ladder (an ellipse), or a person swimming across a river (a parabola).

You'll also see how to draw some impressive screen art using the hyperbola and the ellipse.

SLICING THE CONE

The four curves obtained by slicing through a cone—the circle, ellipse, parabola and hyperbola—are quite distinct. They were first considered in detail by the Greek, Apollonius, around 200 BC.

The starting point is if a pair of intersecting straight lines—like an X—are rotated about an axis of symmetry. This generates a double cone (see the drawings), which can be sliced up in four ways.

If a slice is made at right angles to the axis of symmetry the section is a circle.



A slice taken at an angle between 90° and half the angle between the lines (called the semi-vertical angle of the cone) gives a section called an ellipse.

A slice made at an angle to the axis which equals the semi-vertical angle gives a parabola.

A slice made at an angle less than the semi-vertical angle gives a section with two parts called an hyperbola. There are two parts to the hyperbola because the slice cuts both the upper and lower cones.

There are two special cases. If a slice is taken which includes the axis, that is, the cones are cut in half from top to bottom, then two straight lines are obtained—the ones used to generate the cone in the first place. This is really a special case of the hyperbola. Also if a slice at 90° to the axis is taken between the two cones then all you have is a point, which is just a circle of zero radius.

The drawings below and on the next page should make it clear how all these shapes are obtained. If you like, you can easily make the slices yourself by cutting out a cone from some suitable material—rolled up paper—and then slicing it in different directions. You'll only need a double cone if you want to make a real hyperbola, as this always consists of two parts, but you can model half of it.

DRAWING THE CURVES

All the curves are generated by simple equations, some of which you will already have seen. To enable you to use hi-res graphics commands on the Commodore 64 you need a Simon's BASIC cartridge, or *INPUT*'s machine code utility, starting on page 748. And on the Vic, a Super Expander cartridge.

THE CIRCLE

The equation of a circle is given by

X=A*COS theta
Y=A*SIN theta

where A is the radius, X,Y is a point on the circumference, and theta is the angle made with a fixed line—usually the X axis.

The first program draws a circle with radius A, centred in the middle of the screen:

```
10 CLS
15 LET a=70
25 LET x=a: LET y=0
30 PLOT 127+x,70+y
40 FOR t=0 TO 2*PI STEP .2
50 LET x=a*COS t: LET y=a*SIN t
60 DRAW x-PEEK 23677+127,
   y-PEEK 23678+70
70 NEXT t
```

```
10 HIRES 0,1:COLOUR 1,1
15 A=60
20 C=ATN(1)/45
30 XX=160+A:YY=100
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):Y=A*SIN(TH*C)
60 LINE XX,YY,160+X,100+Y,1
65 XX=160+X:YY=100+Y
70 NEXT TH
80 GOTO 80
```

```
10 GRAPHIC 2:COLOR 6,6,5,5
15 A=200
20 C=ATN(1)/45
25 X=A:Y=0
30 POINT 1,512+X,512+Y
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):Y=A*SIN(TH*C)
60 DRAW 1 TO 512+X,512+Y
70 NEXT TH
80 GOTO 80
```

```
10 MODE 1
15 A=200
20 VDU29,640;512;
25 X=A:Y=0
30 MOVEX,Y
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(RAD(TH)):
   Y=A*SIN(RAD(TH))
60 DRAW X,Y
70 NEXT TH
```

```
10 PMODE4:PCLS:SCREEN1,1
15 A=60
20 C=ATN(1)/45
30 LINE-(127+A,95),PRESET
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):
   Y=A*SIN(TH*C)
60 LINE-(127+X,95+Y),PSET
70 NEXT TH
80 GOTO 80
```

The FOR . . . NEXT loop in Lines 40 to 70 is the part of the program that draws the circle by repeatedly drawing straight line segments at intervals of 10 degrees (or .2 radians). The radius of the circle is set at Line 15.

THE ELLIPSE

The equation for an ellipse is very similar to that for a circle. For an ellipse with major axis 2A and minor axis 2B, the position of any point on the circumference is:

X=A*COS theta
Y=B*SIN theta

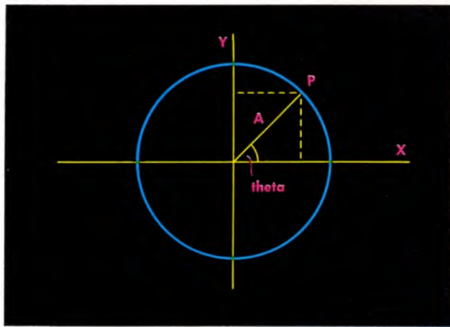
The shape of the ellipse—how squashed it is is determined by A and B. Change these lines:

```
16 LET b=40
50 LET x=a*COS t: LET y=b*SIN t
```

```
16 B=30
50 X=A*COS(TH*C):Y=B*SIN(TH*C)
```

```
16 B=100
50 X=A*COS(TH*C):Y=B*SIN(TH*C)
```

```
16 B=100
50 X=A*COS(RAD(TH)):
   Y=B*SIN(RAD(TH))
```



The circle



```
16 B=30
50 X=A*COS(TH*C): Y=B*SIN
   (TH*C)
```

THE PARABOLA

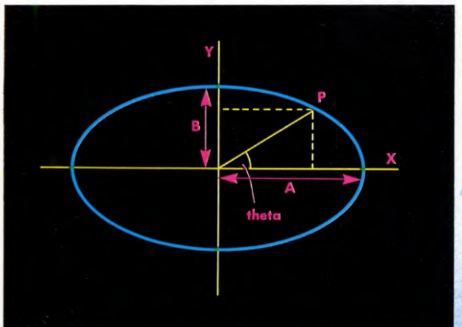
The size of the parabola depends on the value of a variable T, the equations are:

$X=T^2$
$Y=2*T$

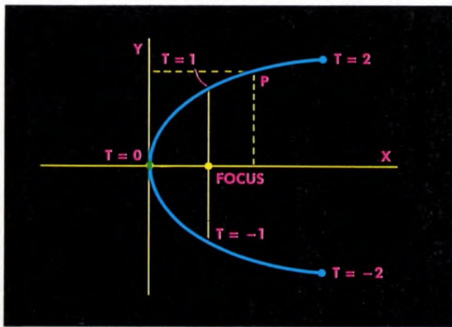
The value T can vary from infinity to minus infinity, but quite a reasonable section of the parabola can be seen between T=2 to T=−2. In the program these values then have to be scaled up by a factor M to fit on the TV screen. These are the programs to draw the parabola:

```
10 CLS
15 LET m=20
25 LET x=4*m: LET y=-4*m
30 PLOT 127+x,80+y
40 FOR t=-2 TO 2 STEP .05
50 LET x=m*t*t: LET y=2*m*t
60 DRAW x-PEEK 23677+127,
   80+y-PEEK 23678+80
70 NEXT t
```

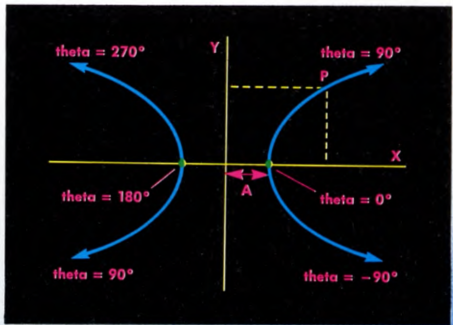
```
10 HIRES 0,1:COLOUR 1,1
15 M=23
```



The ellipse



The parabola



The hyperbola



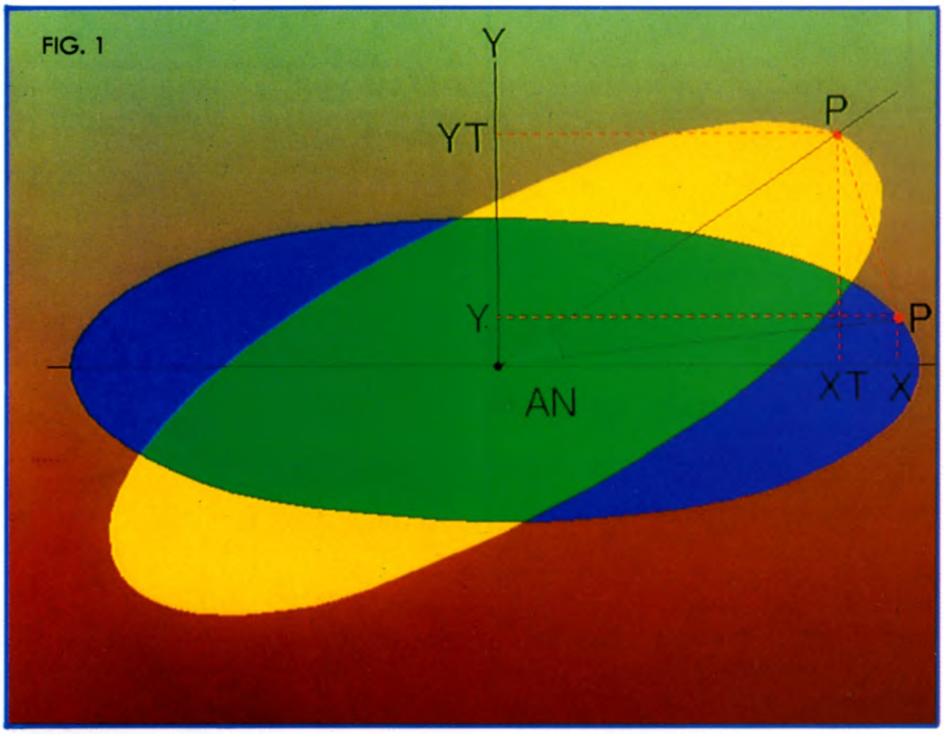


FIG. 1

```
20 C=ATN(1)/45
30 XX=160+M*4:YY=100-4*M
40 FOR T=-2 TO 2 STEP .05
50 X=M*T↑2: Y=2*M*T
60 LINE XX,YY,160+X,100+Y,1
65 XX=160+X:YY=100+Y
70 NEXT T
80 GOTO 80
```

```
10 GRAPHIC 2:COLOR 6,6,5,5
15 M=50
20 C=ATN(1)/45
25 X=M*4:Y=-M*4
30 POINT 1,512+X,512+Y
40 FOR T=-2 TO 2 STEP .05
50 X=M*T↑2:Y=2*M*T
60 DRAW 1 TO 512+X,512+Y
70 NEXT T
80 GOTO 80
```

```
10 MODE 1
15 M=100
20 VDU29,640;512;
25 X=4*M:Y=-4*M
30 MOVE X,Y
40 FOR T=-2 TO 2 STEP .05
50 X=M*T*T:Y=M*2*T
60 DRAW X,Y
70 NEXT T
```

```
10 PMODE4:PCLS:SCREEN1,1
15 M=23
20 C=ATN(1)/45
30 LINE-(127+M*4,95-4*M),PRESET
40 FOR T=-2 TO 2 STEP .05
50 X=M*T↑2:Y=2*M*T
60 LINE-(127+X,95+Y),PSET
70 NEXT T
80 GOTO 80
```

THE HYPERBOLA

The hyperbola equation is:

X=A/COS theta
Y=B*TAN theta

One half of the hyperbola is traced out as theta goes from minus 90° to 90°, and the other half is traced as theta goes from 90° to 270°. It is theoretically possible to use just one loop in the program and take theta from −90° to 270° but there are problems at −90°, 90° and 270° as at these points division by zero occurs, which the computer cannot deal with. Even at values of theta near these, large values are involved. So the program below uses two loops instead. Again, a magnification factor M is used to scale the drawing to fit the screen:

```
10 CLS
15 LET m=30
25 LET x=m/COS-1:LET y=m*TAN-1
30 PLOT 127+x,75+y
40 FOR t=-1 TO 1 STEP .1
50 LET x=m/COS t: LET y=m*TAN t
60 DRAW 127+x-PEEK 23677,
   75+y-PEEK 23678
70 NEXT t
75 LET x=m/COS(PI-1):
   LET y=m*TAN(PI-1)
80 PLOT 127+x, 75+y
90 FOR t=PI-1 TO PI+1 STEP .1
100 LET x=m/COS t: LET y=m*TAN t
110 DRAW 127+x-PEEK 23677,
    75+y-PEEK 23678
120 NEXT t
```

```
10 HIRES 0,1:COLOUR 1,1
15 M=50
20 C=ATN(1)/45
25 X=M/COS(-60*C):Y=M*TAN(-60*C)
30 XX=267:YY=8
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 LINE XX,YY,160+X,100+Y,1
65 XX=160+X:YY=100+Y
70 NEXT TH
75 X=M/COS(120*C):Y=M*TAN(120*C)
80 XX=50:YY=8
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 LINE XX,YY,160+X,100+Y,1
```

FIG. 2



```
115 XX=160+X:YY=100+Y
120 NEXT TH
130 GOTO 130
```

```
10 GRAPHIC 2:COLOR 6,6,5,5
15 M=150
20 C=ATN(1)/45
25 X=M/COS(-60*C):Y=M*TAN(-60*C)
30 POINT 1,512+X,512+Y
40 FOR TH=-60 TO 60 STEP 10
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 DRAW 1 TO 512+X,512+Y
70 NEXT TH
75 X=M/COS(120*C):Y=M*TAN(120*C)
80 POINT 1,INT(512+X),INT(512+Y)
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 DRAW 1 TO 512+X,512+Y
120 NEXT TH
130 GOTO 130
```

```
10 MODE 1
15 M=100
20 VDU 29,640;512;
25 X=M/COS(RAD(-60)):
   Y=M*TAN(RAD(-60))
30 MOVE X,Y
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(RAD(TH)):Y=M*TAN(RAD(TH))
60 DRAW X,Y
70 NEXT TH
80 MOVE -200,-173
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(RAD(TH)):
   Y=M*TAN(RAD(TH))
110 DRAW X,Y
120 NEXT TH
```

```
10 PMODE4:PCLS:SCREEN1,1
15 M=50
20 C=ATN(1)/45
30 LINE-(227,8),PRESET
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 LINE-(127+X,95+Y),PSET
70 NEXT TH
80 LINE-(26,8),PRESET
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 LINE-(127+X,95+Y),PSET
120 NEXT TH
130 GOTO 130
```



ROTATING THE CURVES

The last programs drew the shapes in the simplest possible way with the X axis horizontal and the Y axis vertical. This, though, is not always convenient, and you may need to draw the curves at an angle. **Fig. 1** shows what happens to a point on the edge of an ellipse as it is rotated through an angle of AN degrees. The point P moves from position X,Y to its new position XT,YT and its new coordinates are given by:

XT=X*COS AN−Y*SIN AN
YT=X*SIN AN+Y*COS AN

Here is the rotation routine for each computer:

```
1000 LET xt=x*COS (an*PI/180)-
   y*SIN (an*PI/180)
1010 LET yt=x*SIN (an*PI/180)+
   y*COS (an*PI/180)
1020 RETURN
```

```
1000 XT=X*COS(AN*C)-Y*SIN(AN*C)
1010 YT=Y*COS(AN*C)+X*SIN(AN*C)
1020 RETURN
```

```
1000 DEF PROCrotate
1010 XT=X*COS(RAD(AN))-
   Y*SIN(RAD(AN))
1020 YT=X*SIN(RAD(AN))+
   Y*COS(RAD(AN))
1030 ENDPROC
```

You'll have to make a few alterations to the curve drawing programs to make use of the rotate subroutine. The angle of rotation AN has to be specified (Line 17), the start position has to be rotated, then the lines have to be drawn to the new rotated coordinates XT and YT instead of X and Y. If you like you can alter Line 17 to allow you to INPUT the angle of rotation. On the Dragon and Tandy the INPUT has to come before Line 10.

Here are the changes to make to the ellipse drawing program. Don't forget to add the rotate routine to each program.

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,70+yt
55 GOSUB 1000
```

```
60 DRAW xt-PEEK 23677+127,
   yt-PEEK 23678+70
80 STOP
```

```
17 AN=60
25 X=A:GOSUB 1000
30 XX=160+XT:YY=100+YT
55 GOSUB 1000
60 LINE XX,YY,160+XT,100+YT,1
65 XX=160+XT:YY=100+YT
```

```
17 AN=60
25 X=A:GOSUB 1000
30 POINT 1,512+XT,512+YT
55 GOSUB 1000
60 DRAW 1 TO 512+XT,512+YT
```

```
17 AN=60
28 PROCrotate
30 MOVE XT,YT
55 PROCrotate
60 DRAW XT,YT
80 END
```

```
17 AN=60
25 X=A: GOSUB 1000
30 LINE-(127+XT,95+YT),PRESET
55 GOSUB1000
60 LINE-(127+XT,95+YT),PSET
```

You can use the same subroutine (or PROCedure) to rotate the parabola. Add it to the main program and make the changes given below:

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,80+yt
40 FOR t=-1.75 TO 1.75 STEP .05
55 GOSUB 1000
60 DRAW 127+xt-PEEK 23677,80+
   yt-PEEK 23678
80 STOP
```

```
17 AN=60
28 X=M*4:Y=-M*4:GOSUB 1000
30 XX=160+XT:YY=100+YT
55 GOSUB 1000
60 LINE XX,YY,160+XT,100+YT,1
65 XX=160+XT:YY=100+YT
```

```
17 AN=60
28 GOSUB 1000
30 POINT 1,512+XT,512+YT
55 GOSUB 1000
60 DRAW 1 TO 512+XT,512+YT
```

```
17 AN=60
28 PROCrotate
30 MOVE XT,YT
55 PROCrotate
60 DRAW XT,YT
80 END
```

```
17 AN=60
20 C=ATN(1)/45
25 X=M*4:Y=-M*4:GOSUB1000
30 LINE-(127+XT,95+YT),PRESET
55 GOSUB1000
60 LINE-(127+XT,95+YT),PSET
```

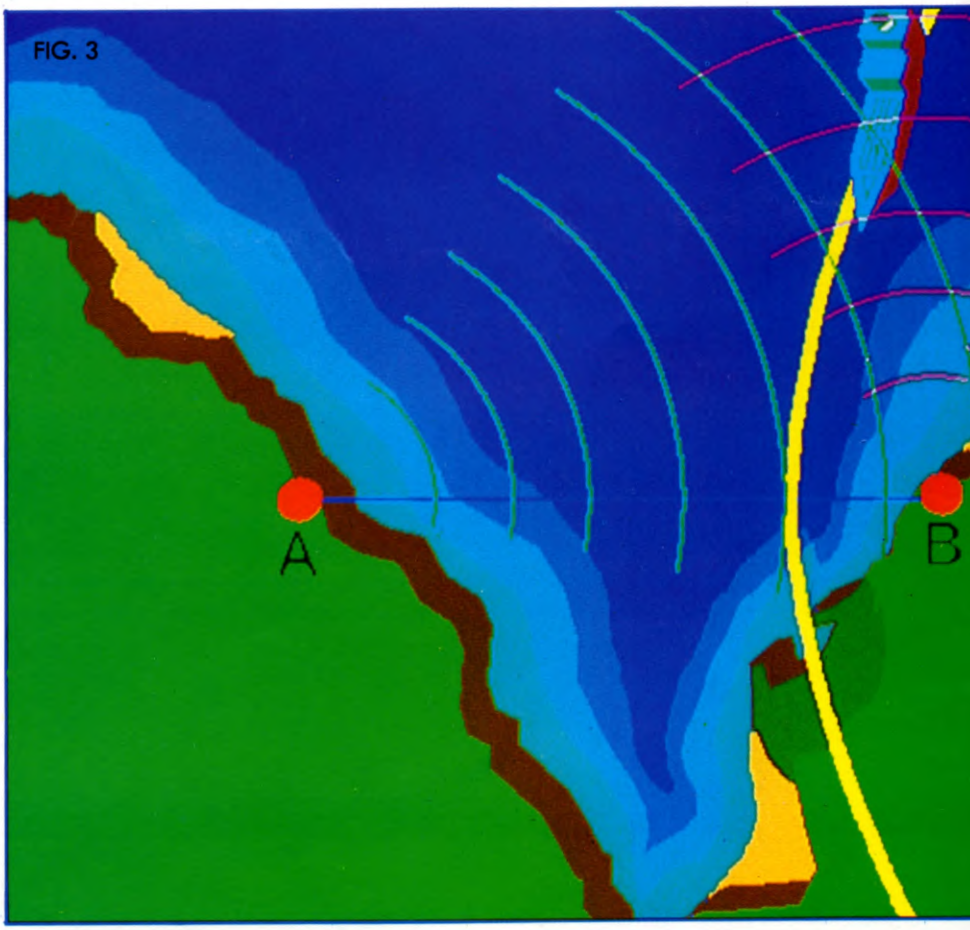
Finally, here are the hyperbola changes:

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,75+yt
55 GOSUB 1000
60 DRAW 127+xt-PEEK 23677,
   75+yt-PEEK 23678
76 GOSUB 1000
80 PLOT 127+xt,75+yt
105 GOSUB 1000
110 DRAW 127+xt-PEEK 23677,
   75+yt-PEEK 23678
130 STOP
```

```
17 AN=60
28 GOSUB 1000
30 XX=160+XT:YY=100+YT
55 GOSUB 1000
60 LINE XX,YY,160+XT,100+YT,1
65 XX=160+XT:YY=100+YT
78 GOSUB 1000
80 XX=INT(160+XT):YY=INT(100+YT)
105 GOSUB 1000
110 LINE XX,YY,160+XT,100+YT,1
115 XX=160+XT:YY=100+YT
```

```
17 AN=60
28 GOSUB 1000
30 POINT 1,512+XT,512+YT
55 GOSUB 1000
```

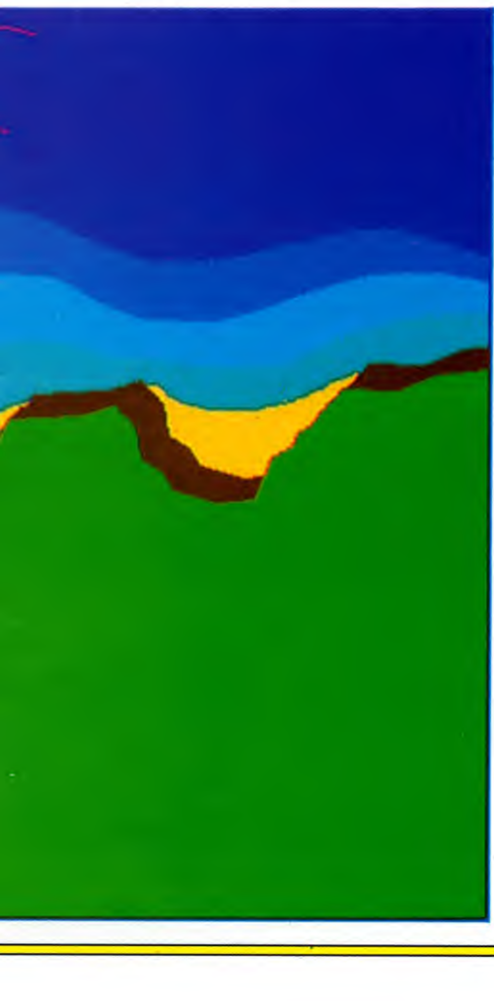
FIG. 3



```
60 DRAW 1 TO 512+XT,512+YT
78 GOSUB 1000
80 POINT 1,INT(512+XT),
   INT(512+YT)
105 GOSUB 1000
110 DRAW 1 TO 512+XT,512+YT
```

```
17 AN=60
28 PROCrotate
30 MOVE XT,YT
55 PROCrotate
60 DRAW XT,YT
80 X=M/COS(RAD(120)):
   Y=M*TAN(RAD(120))
82 PROCrotate
85 MOVE XT,YT
105 PROCrotate
110 DRAW XT,YT
140 END
```

```
17 AN=60
25 X=M/COS(-60*C):Y=M*TAN
   (-60*C):GOSUB1000
30 LINE-(127+XT,95+YT),PRESET
55 GOSUB1000
60 LINE-(XT+127,YT+95),PSET
75 X=M/COS(135*C):Y=
   M*TAN(135*C):GOSUB1000
80 DRAW"BM"+STR$(INT(127+XT))+
   ","+STR$(INT(95+YT))
90 FORTH=135 TO 240STEP5
105 GOSUB1000
110 LINE-(127+XT,95+YT),PSET
```



PRACTICAL APPLICATIONS

All these curves can be used in some practical way.

The circle has so many uses that it is impossible to list them all. The wheel is an obvious example of a circle, and ball bearings are an obvious use of a sphere. Spheres, or approximations to them, often occur in nature. Examples range from water droplets to peas to planets. But the spheres are very rarely perfect due to effects of gravity, wind or other forces. A planet revolving round a star could move in a circular orbit, although this is more likely to be elliptical.

One useful practical application of the circle is in working out the lowest transport costs for something that can be bought from one of two distribution depots. For example, suppose you wanted to buy a computer which can be supplied by either of two firms A or B which are 300 miles apart. Say firm A sends the computers by special carriers at the rate of 10p a mile while firm B sends the computers by its own van at 5p per mile. It is very straightforward to mark out the area on a map where it is cheaper to buy from A or B. The idea is to mark all points where the two costs are equal, and join them up by a line. In this case you can afford to have something delivered twice as far from depot B as they only charge half as much. So you should mark all points where the distance from B is twice the distance from A.

One point is on a line between A and B, 100 miles from A and 200 miles from B (since 100 × 10p equals 200 × 5p). Another point is on the same line 300 miles from A in the opposite direction to B (300 × 10p equals 600 × 5p). If you join up all these points the line traced out is a circle with a radius of 200 miles as shown in **fig. 2.** If you live inside the circle it is cheaper to buy from A, and if you live outside it is cheaper to buy from B.

The ellipse has practical uses too. If you project the shadow of an ellipse on to a flat surface then it is possible to hold the ellipse at one particular angle where its shadow is a perfect circle. The property is made use of in valves in circular ducts, where an elliptical flap can be used to control air or gas flow. The flap fits the pipe exactly when it reaches the correct angle and so blocks off the pipe.

The parabola, of course, describes the curve traced out by a projectile (see pages 740 to 747). Comets can also travel in a parabolic path round the Sun.

A very useful property of the parabola is that rays of light, heat or anything else parallel to the axis are reflected through the focus. This property works in both directions so an electric bulb placed at the focus will produce a parallel beam of light, as used in car headlights. In the other direction parallel rays from the Sun can be concentrated at the focus to produce very high temperatures as in a solar furnace.

In practice, the reflectors used for these purposes are three-dimensional paraboloids. A further use of paraboloids is in radio or radar aerial dishes where the aerial element is placed at the focus and can be used for both transmitting and receiving signals.

An important feature of the hyperbola is that it consists of two parts. And a practical use is in a system of radar navigation for ships. The system relies on two radar stations. One station transmits signals normally, and the other simply retransmits signals received from the first station. Any ship in the vicinity receives both signals and notes the time difference in their arrival. If it moves so as to keep this time difference constant then it will follow a hyperbolic path as shown in **fig. 3.** If the ship also receives signals from two other radar stations and again notes the time difference this will give a second hyperbola and the intersection of the two gives the position of the ship. There is no confusion over which branch of the hyperbola the ship is on as the signal which arrives first can be detected.

In the next article, you'll see how to program the computer to demonstrate some practical uses for conic sections.

TROUBLE SHOOTER

- The programs in this article have been designed to make the best use of the TV screen. When you use them in your own programs you'll have to change the magnification factor M so the curves are drawn to the correct size.
- You must also take care with the rotated parabola and hyperbola to make sure that the ends of the curves stay within the screen. (This doesn't apply to the Acorns as these can 'draw' off the screen quite happily.) To prevent this, alter the ends of the FOR ... NEXT loops in Line 40 of the parabola program and Lines 40 and 90 of the hyperbola program. You'll have to find out the exact limits by trial and error.

THE MIDAS TOUCH

It's time to get rich quick. But do you invest in new technology before exploring? How do you interpret the result? And when do you sell? You'll have to be shrewd in Goldmine

In the first part of this game, you saw how to set up the various options available to the player—Research and Development, Exploration and Report, Increasing Mine Depth, and Exchanging Gold for Dollars. Now complete your Goldmine program with the subroutines which handle each of these options.

Research and Development follows the player choosing option 1, Exploration and Report is option 2, Increasing Mine Depth is option 3, and Exchanging Gold for Dollars is option 4. Option 5 is Pass, so no complete subroutine is needed. Options 1, 2 and 4 introduce the elements of randomness needed to make the game parallel the real world.

RESEARCH AND DEVELOPMENT

```
1ØØØ BORDER 6: PAPER 6: INK Ø: CLS
1Ø1Ø PRINT PAPER 1; INK 6;AT 3,4;
   "□RESEARCH & DEVELOPMENT□";
   AT 4,4;"(to lower mining costs) "
1Ø2Ø PRINT AT 7,6;"How much would
   you";TAB 5;"like to invest ? ($)": INPUT rd
1Ø5Ø LET a(m,4)=a(m,4)-INT (rd*.Ø5)-1
1Ø6Ø IF a(m,4)<Ø THEN LET a(m,4)=Ø
1Ø8Ø LET a(m,2)=a(m,2)-rd: LET
   a(m,1)=a(m,1)-rd
11ØØ PRINT AT 13,3;"Your mining costs will
   be";TAB 3;"reduced by $";INT
   (rd*.Ø5)+1;"□per 2ØØm"
111Ø FOR z=1 TO 3ØØ: NEXT z
112Ø RETURN
```

```
1ØØØ POKE5328Ø,7:POKE53281,7:
   PRINT"■■"
1Ø1Ø PRINT"■RESEARCH AND
   DEVELOPMENT":PRINT"(TO LOWER
   MINING COSTS)"
1Ø2Ø PRINT "■■■■■HOW MUCH
   WOULD YOU LIKE TO INVEST
   ($)":INPUTRD
1Ø3Ø R1=INT(RD*.5)-1
1Ø5Ø A(M,4)=A(M,4)-R1
1Ø6Ø IFA(M,4)<ØTHENA(M,4)=Ø
1Ø8Ø A(M,2)=A(M,2)-RD:A(M,1)
   =A(M,1)-RD
11ØØ PRINT"■■■■■■■■■
   YOUR MINING COSTS WILL BE REDUCED
   BY:":PRINT"$";R1+1;
111Ø PRINT "PER 2ØØM":FORZ=1
   TO23ØØ:NEXT
112Ø RETURN
```

```
1ØØØ PRINT"■■"
1Ø1Ø PRINT"■RESEARCH,
   DEVELOPMENT":PRINT"■■
   TO LOWER MINING COSTS"
1Ø2Ø RD=Ø:PRINT"■■■■■■
   HOW MUCH WOULD YOU□□□□LIKE
   TO INVEST ($)":INPUTRD
1Ø3Ø R1=INT(RD*.5)-1
1Ø5Ø A(M,4)=A(M,4)-R1
1Ø6Ø IFA(M,4)<ØTHENA(M,4)=Ø
1Ø8Ø A(M,2)=A(M,2)-RD:A(M,1)
   =A(M,1)-RD
11ØØ PRINT"■■■■■YOUR
   MINING COSTS WILL BE REDUCED
   BY:":PRINT"$";R1+1"PER 2ØØM"
111Ø FORZ=1TO23ØØ:NEXT
112Ø RETURN
```

```
1ØØØ COLOUR129:COLOUR2:CLS
1Ø1Ø PRINTTAB(8,3)"RESEARCH AND
   DEVELOPMENT"TAB(9,5)"(LOWERS
   MINING COSTS)"
1Ø2Ø PRINT'"HOW MUCH WOULD YOU LIKE
   TO INVEST ($)":INPUTRD
1Ø5Ø A(M,4)=A(M,4)-INT(RD*.Ø5)
1Ø6Ø IF A(M,4)<Ø THEN A(M,4)=Ø
1Ø8Ø A(M,2)=A(M,2)-RD:A(M,1)
   =A(M,1)-RD
11ØØ PRINTTAB(Ø,13)"YOUR MINING COSTS
   WILL BE REDUCED BY □□□□□$";
   INT(RD*.Ø5);"□PER 2ØØm"
111Ø FOR Z=1 TO 4ØØØ:NEXT
112Ø RETURN
```

```
1ØØØ CLS
1Ø1Ø PRINT@3,"research and
   development":PRINT@35,"(TO
   LOWER MINING COSTS)"
1Ø2Ø PRINT:INPUT"HOW MUCH WOULD YOU
   LIKE TO□□□□□□□INVEST ($)";RD
1Ø3Ø IF RD<Ø THEN 1ØØØ
1Ø5Ø A(M,3)=A(M,3)-INT(RD/2Ø)-1
1Ø6Ø IF A(M,3)<Ø THEN A(M,3)=Ø
1Ø8Ø A(M,1)=A(M,1)-RD:A(M,Ø)
   =A(M,Ø)-RD
11ØØ PRINT@257,"YOUR MINING
   COSTS WILL BE□□□□□□□□
   REDUCED BY $";INT(RD/2Ø)+1;
   "PER 2ØØm"
111Ø FORZ=1TO2ØØØ:NEXT
112Ø RETURN
```

In the Spectrum, Commodore 64 and Acorn programs, Line 1ØØØ sets up the screen colours and clears the screen. In the Vic 20 and Dragon/Tandy programs, the screen colour remains the same and the display is simply cleared. Line 1Ø1Ø prints up the heading on the screen, before Line 1Ø2Ø asks the player how much money should be invested—RD (rd in the Spectrum program) is the amount chosen.

Line 1Ø5Ø decreases the mining cost an amount related to the amount of money spent on research and development. Line 1Ø6Ø makes sure the mining costs do not become negative. Line 1Ø8Ø adjusts the cash assets and total assets to take account of the amount invested in R & D.

The amount by which mining costs have been reduced is displayed by Line 11ØØ (and Line 111Ø, in the case of the Commodores). Line 111Ø contains a FOR . . . NEXT loop to put in a short delay before the subroutine ends.

EXPLORATION AND REPORT

```
2ØØØ PAPER 4: BORDER 4: INK Ø: CLS
2Ø3Ø LET r(m)=Ø: LET c(m,1)=INT
   (RND*9Ø)+1Ø: LET c(m,2)=INT
   ((RND*5)+2)*2ØØ: LET c(m,3)=INT
   (RND*2ØØ)+1: LET ll=INT (RND*3)-1
2Ø5Ø LET c(m,4)=c(m,2)+ll*2ØØ
2Ø7Ø LET c(m,5)=Ø: LET kk=INT (RND*
   1ØØ): IF kk<c(m,1) THEN LET c(m,5)=1
2Ø8Ø PRINT PAPER 6; INK Ø;AT 2,6;"□
   SCIENTIFIC REPORT□": PRINTAT 5,5;
   "Chance of gold=□";c(m,1);"%":
   PRINT AT 7,5;"Expected Depth=□";
   c(m,2);"m": PRINT AT 9,5;"Expected
   amount=□";c(m,3);"kg"
21ØØ LET z=INT (RND*15ØØØØ): LET a(m,2)
   =a(m,2)-z: LET a(m,1)=a(m,1)-z
```



```
2110 PRINT FLASH 1;AT 12,0;"Would you like
   to mine? (y or n)"
2120 LET r$=INKEY$: IF r$="" THEN
   GOTO 2120
2130 IF r$="y" THEN LET a(m,6)=0: LET
   r(m)=1: GOTO 3000
2500 RETURN
```

```
2000 POKE53280,5:POKE53281,5:
   PRINT"♥■";
2030 R(M)=0:C(M,1)=INT(RND(1)*90)+
   10:C(M,2)=INT((RND(1)*5)+2)*200
2031 C(M,3)=INT(RND(1)*200)+1:
   LL=INT(RND(1)*3)-1
2050 C(M,4)=C(M,2)+LL*200
2070 C(M,5)=0:KK=INT(RND(1)*
   100):IFKK<C(M,1)THENC(M.5)=1
2080 PRINT"■■{DOWN}{DOWN}{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}
   {RIGHT}□SCIENTIFIC REPORT□"
2081 PRINT"{DOWN}{DOWN}{DOWN}{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}
   CHANCE OF GOLD=";C(M,1)"%"
2082 PRINT"{DOWN}{DOWN}{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}
   EXPECTED DEPTH=";C(M,2);"M"
2083 PRINT"{DOWN}{DOWN}{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}
   EXPECTED AMOUNT";C(M,3) "KG"
2100 Z=INT(RND(1)*150000):A(M,2)
   =A(M,2)-Z:A(M,1)=A(M,1)-Z
2110 PRINT"{DOWN}{DOWN}{DOWN}■WOULD YOU LIKE
   TO MINE (Y OR N)?"
2120 GETR$:IFR$<>"Y"ANDR$<>
   "N"THEN2120
2130 IFR$="Y"THENA(M,6)=0:R(M)
   =1:GOTO3000
2500 RETURN
```

```
2000 PRINT"♥■"
2030 R(M)=0:C(M,1)=INT(RND(1)*90)
   +10:C(M,2)=INT((RND(1)*5)+2)*200
2031 C(M,3)=INT(RND(1)*200)+1:
   LL=INT(RND(1)*3)-1
2050 C(M,4)=C(M,2)+LL*200
2070 C(M,5)=0:KK=INT(RND(1)*
   100):IFKK<C(M,1)THENC(M,5)=1
2080 PRINT"■■{RIGHT}{RIGHT}SCIENTIFIC REPORT"
2081 PRINT"■{DOWN}{DOWN}{DOWN}CHANCE OF
   GOLD=";C(M,1)"%"
2082 PRINT"{DOWN}{DOWN}EXPECTED DEPTH
   =";C(M,2);"M"
2083 PRINT"{DOWN}{DOWN}EXPECTED AMOUNT";
   C(M,3)"KG"
2100 Z=INT(RND(1)*150000):A(M,2)
   =A(M,2)-Z:A(M,1)=A(M,1)-Z
2110 PRINT"{DOWN}{DOWN}{DOWN}■WOULD YOU LIKE TO
   MINE{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}{RIGHT}■(Y OR N)?"
2120 GETR$:IFR$<>"Y"ANDR$<>
   "N"THEN2120
2130 IFR$="Y"THENA(M,6)=0:R(M)
   =1:GOTO3000
2500 RETURN
```





```
2ØØØ COLOUR129:COLOUR3:CLS
2Ø3Ø R(M)=Ø:C(M,1)=RND(9Ø)+9:
    C(M,2)=RND(5)*2ØØ+4ØØ:C(M,3)
    =RND(2ØØ):LL=RND(3)-2
2Ø5Ø C(M,4)=C(M,2)+LL*2ØØ
2Ø7Ø C(M,5)=Ø:KK=RND(1ØØ):IF
    KK<C(M,1) THEN C(M,5)=1
2Ø8Ø PRINTTAB(1Ø,3)"SCIENTIFIC
    REPORT"TAB(1Ø,1Ø)"CHANCE OF
    GOLD=□";C(M,1);"%"TAB(1Ø,12)
    "EXPECTED DEPTH=□";C(M,2);
    "m"TAB(1Ø,14)"EXPECTED
    AMOUNT=□";C(M,3);"KG"
21ØØ Z=RND(15ØØØØ):A(M,2)=
    A(M,2)-Z:A(M,1)=A(M,1)-Z
211Ø PRINTTAB(5,2Ø)"WOULD YOU LIKE TO
    MINE (Y/N)?"
212Ø R$=GET$
213Ø IF R$="Y" THEN A(M,6)=Ø:
    R(M)=1:GOTO 3ØØØ
25ØØ RETURN
```

```
2ØØØ CLS
2Ø3Ø R(M)=Ø:C(M,Ø)=RND(9Ø)+9:
    C(M,1)=(RND(5)+1)*2ØØ:C(M,2)
    =RND(2ØØ):LL=RND(3)-2
2Ø5Ø C(M,3)=C(M,1)+LL*2ØØ
2Ø7Ø C(M,4)=Ø:KK=RND(1ØØ)-1:
    IF KK<C(M,Ø) THEN C(M,4)=1
2Ø8Ø PRINT@6,"scientific report":
    PRINT@129,"CHANCE OF GOLD=";
    C(M,Ø);"%":PRINT@193,
    "EXPECTED DEPTH =";C(M,1);
    "m":PRINT@257,"EXPECTED
    AMOUNT=";C(M,2);"kg"
21ØØ Z=RND(15ØØØØ)-1:A(M,1)=
    A(M,1)-Z:A(M,Ø)=A(M,Ø)-Z
211Ø PRINT@353,"WOULD YOU LIKE TO
    MINE (Y/N) ?"
212Ø R$=INKEY$:IF R$<>"Y" AND
    R$<>"N" THEN 212Ø
213Ø IF R$="Y" THEN A(M,5)=Ø:
    R(M)=1:GOTO3ØØØ
25ØØ RETURN
```

All machines clear the screen in Line 2ØØØ. The Spectrum, Commodore 64, and Acorn programs also change the screen colours. Line 2Ø3Ø sets R(M) (r(m), in the case of the Spectrum) to zero to indicate that excavation hasn't yet started. The line also chooses the chance of finding gold, expected depth and the expected amount. LL (or ll) is a random number between −1 and 1 which is used in the next line to determine the actual depth of the gold—remember, the value in C(M,2) is just an expected depth.

Line 2Ø5Ø sets C(M,4) equal to the value of

C(M,2) plus or minus 200 metres—200 times LL. Next, Line 2070 decides if the mine actually contains any gold. C(M,5) is set to zero to indicate there's no gold. KK is a random number between zero and 99. KK is compared with the chance of finding gold—if KK is less, then C(M,5) is set to one to indicate that there is gold in the mine.

Line 2080 presents the player with the Scientific Report on the mine—the Commodores use Lines 2080 to 2083. Although the player is told what chance there is of finding gold and the expected depth, whether it is actually there or not is controlled by the various random factors. So you need to use your judgement about whether the investment is worthwhile.

Now for the bad news: the report has to be paid for. It's impossible to predict how much the report will cost, but it may cost anything between nothing and $150,000—the value of Z chosen in Line 2100. The cost of the exploration and report is subtracted from the cash assets and this deduction appears in the total assets.

Now the player is given the opportunity to start excavations—Line 2110 asks WOULD YOU LIKE TO MINE? If the answer is yes, then the program jumps to the mining routine starting at Line 3000.

EXCAVATION

```
3000 BORDER 6: PAPER 6: INK 1: CLS
3010 IF r(m)=0 THEN PRINT FLASH 1;AT
   9,2;"You have not explored yet!": FOR z=1
   TO 10: BEEP .3,-10: NEXT z: RETURN
3020 BORDER 5: INK 0: PAPER 4: CLS
3022 PRINT PAPER 5;TAB 14;CHR$ 147;CHR$
   148;CHR$ 149;TAB 14;CHR$ 150;CHR$
   151;CHR$ 152;CHR$ 153;TAB 13;CHR$
   154;CHR$ 155;CHR$ 156;CHR$ 157;CHR$
   158;TAB 31;CHR$ 32
3025 FOR z=1 TO 32: PRINT CHR$ 144;:
   NEXT z
3060 PRINT AT 4,0;: FOR z=100 TO 1400
   STEP 100: PRINT TAB 4-LEN STR$ z;z:
   NEXT z
3090 LET a(m,2)=a(m,2)-a(m,4): LET
   a(m,1)=a(m,1)-a(m,4): LET
   a(m,6)=a(m,6)+200: PAUSE 30
3100 PRINT AT 3,15;CHR$ 146: FOR f=4 TO
   (a(m,6)/100)+3: PRINT AT f,15;CHR$
   145: FOR w=1 TO 10: BEEP .01,-20:
   NEXT w: NEXT f
3120 IF a(m,6)=c(m,4) AND c(m,5)=1
   THEN GOTO 3500
3130 PRINT FLASH 1; PAPER 5;AT 6,18;"No
   gold yet!": IF a(m,6)=c(m,2)+200 THEN
   PRINT FLASH 1; PAPER 1; INK 6;AT
   18,0;"This mine doesn't contain any gold.
   Try starting another one.": FOR z=1 TO 10:
   BEEP .5,-20: NEXT z: LET a(m,6)=0:
   LET r(m)=0
3140 PAUSE 150
3300 RETURN
3500 PRINT PAPER 6; INK 2; FLASH 1;AT
   f,12;"G O L D": FOR z=-20 TO 50:
   BEEP .017,z: NEXT z: PAUSE 75
3550 LET a(m,5)=a(m,5)+1: LET
   a(m,3)=a(m,3)+c(m,3): LET
   a(m,1)=a(m,1)+(a(m,3)*er):
   LET a(m,6)=0: LET r(m)=0: GOTO 3300
```

```
3000 POKE53280,7:POKE53281,7
3010 IFR(M)<>0THEN3020
3011 PRINT"■■■■■■■■■■
   ■■■■■YOU HAVE NOT EXPLORED
   YET!":FORZ=1TO2300:NEXT:RETURN
3020 POKE53280,3:POKE53281,5:
   PRINT"■■"
3022 PRINTTAB(14);"□□□":PRINT
   TAB(14);"□□□□□":PRINTTAB
   (13);"□□□□□"
3025 FORZ=0TO39:PRINT"■";:NEXT
3060 PRINT"■■■■■■■";:
   FORZ=100TO1400STEP100:PRINT
   TAB(5-LEN(STR$(Z)));Z:NEXT
3090 A(M,2)=A(M,2)-A(M,4):A(M,1)=
   A(M,1)-A(M,4):A(M,6)=A(M,6)+200
3095 FORF=0TO90:NEXT
3100 PRINT"■■■■■■■■■■■
   ■■■■■■■■■■■■■■"
3101 PRINT"■■■■■■■■■■
   ■■■■■■■■■"
3102 FORF=2TO A(M,6)/100:PRINT
   "■■■■■■■■■■■■■
   ■■■■■□"
3104 POKE54272,33:POKE54273,33:
   POKE54277,15:POKE54296,15
3105 POKE54276,129:FORZ=1TO240: NEXT
3110 NEXT:POKE54296,0
3120 IFA(M,6)=C(M,4)ANDC(M,5)
   =1THEN 3500
3130 PRINT"■■■■■■■■
   ■■■■■■■■■■■
   ■■■■■■■■■■■■■
   NO GOLD YET!!!"
3131 IFA(M,6)<>C(M,2)+200THEN 3140
3132 PRINT"■■■THIS MINE DOESN'T
   CONTAIN ANY GOLD. TRY□□STARTING
   ANOTHER ONE."
3134 A(M,6)=0:R(M)=0
3140 FORF=1TO2500:NEXT
3300 RETURN
3500 PRINT "■■■■":FORZ=1
   TOA(M,6)/100:PRINT:NEXT
3505 PRINT"■■■■■■■■■■
   ■■■■□G□O□L□D□"
3510 FORF=54272TO54296:POKEF,0:NEXT
3520 POKE54284,15:POKE54283,17:
   POKE54296,14
3530 FORF=64TO124
3540 POKE54280,F:FORG=1TO20:
   NEXT:NEXT
3550 FORF=124TO64STEP-1:POKE
   54280,F:FORG=1TO20:NEXT:NEXT
3560 POKE54296,0
3570 A(M,5)=A(M,5)+1:A(M,3)=A(M,3)
   +C(M,3):A(M,1)=A(M,1)+A(M,3)*ER
3580 A(M,6)=0:R(M)=0:GOTO3300
```

```
3000 POKE 36879,25
3010 IFR(M)<>0THEN3020
3011 PRINT"■■■■■■■■■■
   ■YOU HAVE NOT EXPLORED YET!":
   FORZ=1TO2300:NEXT:RETURN
3020 PRINT"■■■"
3022 PRINTTAB(14);"■■■":
   PRINTTAB(14);"■■■□■■■"
3025 FORZZ=1TO15:FORZ=0TO20:
   PRINT"■□";:NEXT:PRINT:NEXT
3060 PRINT"■■■■■■";:FOR Z=
   100TO1400STEP100:PRINT"■"Z: NEXT
3090 A(M,2)=A(M,2)-A(M,4):A(M,1)
   =A(M,1)-A(M,4):A(M,6)=A(M,6)
   +200
3095 FORF=0TO90:NEXT
3100 PRINT"■■■■■"SPC(15)
   "■□"
3102 FORF=2TO A(M,6)/100:PRINT
   SPC(15)"■□":POKE 36877,128+F*3
3104 FORDE=5TO15STEP.3:POKE36878,
   DE:NEXT:POKE36877,0
3110 NEXT
3120 IFA(M,6)=C(M,4)ANDC(M,5)
   =1THEN 3500
3130 PRINT"■■■■■■■
   ■■■■■■■■■■■■
   ■■■■■■■NO GOLD YET!!!"
3131 IFA(M,6)<>C(M,2)+200THEN3140
3132 PRINT"■■■THIS MINE
   DOESN'T□□□CONTAIN ANY GOLD.
   TRY□□STARTING ANOTHER ONE."
3134 A(M,6)=0:R(M)=0
3140 FORF=1TO3000:NEXT
3300 RETURN
3500 PRINT "■■■":FORZ=1TO
   A(M,6)/100:PRINT:NEXT
3505 PRINT"■■■■■■■■■■
   ■■■■□G□O□L□D□"
3508 FORDE=250TO127STEP-1:POKE
   36876,DE:NEXT
3510 FORG=1TO2000:NEXT
3570 A(M,5)=A(M,5)+1:A(M,3)=A(M,3)
   +C(M,3):A(M,1)=A(M,1)+A(M,3)*ER
3580 A(M,6)=0:R(M)=0:GOTO3300
```

```
3000 COLOUR130:COLOUR0:CLS
```

```
3010 IF R(M)=0 THEN PRINTTAB(6,12)
    "YOU HAVE NOT EXPLORED YET!":FOR Z=
    1 TO 10:SOUND1,-15,100,1:SOUND1,0,
    1,1:NEXT:Z=INKEY(300):RETURN
3020 CLS
3022 VDU 31,16,3,224,225,226,31,16,4,227,
    228,229,230,31,15,5,231,232,233,234,235
3025 PRINT:FOR Z=1 TO 40:VDU236:NEXT
3060 PRINT:FOR Z=100 TO 1400 STEP
    100:PRINTTAB(4-LENSTR$Z);Z:NEXT
3090 A(M,2)=A(M,2)-A(M,4):A(M,1)
    =A(M,1)-A(M,4):A(M,6)=A(M,6)+
    200:Z=INKEY(60)
3100 VDU31,17,6,238:FOR F=7 TO
    (A(M,6)/100)+7:VDU31,17,F-1,237:
    FOR Z=1 TO 13: SOUND0,-15,6,1:
    SOUND0,0,0,1:NEXT:SOUND16,0,0,1:NEXT
3120 IF A(M,6)=C(M,4) AND C(M,5)
    =1 THEN 3500
3125 COLOUR1
3130 PRINTTAB(20,10)"NO GOLD YET!":IF
    A(M,6)=C(M,2)+200 THEN COLOUR3:
    PRINTTAB(0,29)"THIS MINE DOESN'T
    CONTAIN ANY GOLD. TRY STARTING
    ANOTHER ONE.":A(M,6)=0:R(M)=0
3140 FOR Z=1 TO 4000:NEXT
3300 RETURN
3500 COLOUR1:PRINTTAB(14,F)"G□O□L
    □D":FOR Z=0 TO 250 STEP 10:
    SOUND1,-15,Z,1:NEXT:Z=INKEY (150)
3550 A(M,5)=A(M,5)+1:A(M,3)=
    A(M,3)+C(M,3):A(M,1)=A(M,1)+(A(M,
    3)*ER):A(M,6)=0:R(M)=0:GOTO 3300
```

```
3000 CLS
3010 IF R(M)=0 THEN PRINT@66,"YOU
HAVE NOT EXPLORED YET !";:FORZ=1
    TO10:SOUND120,1:NEXT:RETURN
3015 IF A(M,5)>0 THEN LINE(140,40)
    -(157,191),PRESET,BF:GOTO3090
3020 PCLS:SCREEN1,0:COLOR3:LINE
    (0,0)-(255,31),PSET,BF
3022 PUT(131,8)-(168,31),H,PSET
3025 FORZ=0TO 31: PUT(Z*8,32)-
    (Z*8+7,34),T,PSET:NEXT
3060 FORZ=100 TO 1400 STEP 100:
    Z$=STR$(Z)+"-":DRAW"C4S8BM"
    +STR$(49-8*LEN(Z$))+","+STR$
    (32+10*Z/100):GOSUB9000:NEXT
3090 SCREEN1,0:A(M,1)=A(M,1)-
    A(M,3):A(M,0)=A(M,0)-A(M,4):
    A(M,5)=A(M,5)+200
3100 PUT(145,32)-(152,39),D,PSET:
    FORF=4TO(A(M,5)/100)+3:PUT(145,
    F*10)-(152,F*10+9),B,PSET:
    PLAY"T5O01BDBDEBDBDE":NEXT
3120 IF A(M,5)=C(M,3) AND C(M,4)
    =1 THEN 3500
3125 FORZ=1TO1000:NEXT
3130 PRINT@40," NO GOLD YET ! ";:
    IF A(M,5)=C(M,1)+200 THEN
    PRINT@128," THIS MINE DOESN'T
    CONTAIN ANY□□□GOLD.□TRY
    STARTING ANOTHER ONE.□";:PLAY
    "T5O03CDEFG":A(M,5)=0: R(M)=0
3140 FORZ=1TO2500:NEXT
3300 RETURN
3500 F=40+A(M,5)/10:COLOR2:LINE
    (140,F)-(157,F+5),PSET,BF
3510 FORZ=1TO10:PLAY"T5O2CA":PUT
    (140,F)-(157,F+5),H,NOT:NEXT
3520 FORZ=1TO2000:NEXT
3550 A(M,4)=A(M,4)+1:A(M,2)=A(M,2)
    +C(M,2):A(M,0)=A(M,0)+(A(M,2)
    *ER):A(M,5)=0:R(M)=0:GOTO3300
```

This routine is called from two places within the program. As you have already seen, you are given the option of mining from within the Exploration and Report subroutine, but it is also used when you opt to increase the mine depth by 200 metres—choice 3 on the list.

As usual, the first line in the routine simply clears the screen, or sets up the screen colours and clears the screen. Line 3010 checks that the exploration phase has been completed before excavation can begin. In the case of the Commodores, Lines 3010 and 3011 check for exploration and display the appropriate message. Line 3020 prepares the screen for the display again.

Lines 3022 to 3090 display the graphics which show the goldmine on screen. Line 3100 illustrates the excavation and makes some sound effects. The Commodore programs use Lines 3100 to 3110 to show the excavation and make the sound effects.

Line 3120 checks if the excavation has reached the level of the gold, and that there is gold in the mine (it's possible to reach the expected level of the gold, only to find that there is none after all in the mine). If gold has been reached, the program jumps to Line 3500 which tells the player that gold has been discovered, and plays a tune—the Commodore programs, again, spread the commands across more than one line. Line 3550 adjusts the value of the player's assets, according to the value of the discovered gold.

If no gold has been discovered, the program continues to Line 3130. If the excavation has passed the expected gold level by 200 metres, the player is told that the mine has no gold. If the excavation hasn't got that far, the player is told NO GOLD YET!

```
4000 PAPER 6: INK 1: BORDER 6: CLS
4020 PRINT INVERSE 1;AT 2,7;"□EXCHANGE
    AGENCY□":PRINT AT 6,0;"The current
    exchange rate is:—";AT 8,5;"1 kg of
    gold=□$";er;AT 12,2;"Enter no. of kg to
    exchange": INPUT nte
4070 IF nte>a(m,3) THEN PRINT FLASH 1;AT
    16,0;"You do not have that much gold!"
4080 LET nte=INT nte
4090 IF nte>a(m,3) OR nte<0 THEN GOTO
    4020
4095 PRINT AT 16,0;CHR$ 32;TAB 31;CHR$ 32
4100 LET a(m,3)=a(m,3)-nte: LET
    a(m,2)=a(m,2)+(nte*er): LET
    a(m,1)=a(m,1)+(nte*er)
4130 PRINT PAPER 5;AT 16,1;nte;"kg
    exchanged for□$";nte*er: PAUSE 170:
    RETURN
5000 RETURN
```

```
4000 POKE53280,7:POKE53281,7:
    PRINT"■■"
4020 PRINT"□□□□□□□■■■
    □EXCHANGE AGENCY□"
4030 PRINT"■■■■THE CURRENT
    EXCHANGE RATE IS:—"
4040 PRINT:PRINT"■■■1 KG OF
    GOLD=$"ER
4050 PRINT:PRINT:PRINT"ENTER NO. OF KG
    TO EXCHANGE":INPUT NT
4060 IFNT>A(M,3)THENPRINT"■YOU DO
    NOT HAVE THAT AMOUNT OF GOLD!!!!!"
4070 NT=INT(NT)
4090 IFNTE>A(M,3)ORNTE<0THEN 4020
4100 A(M,3)=A(M,3)-NT:A(M,2)=A(M,2)
    +(NT*ER):A(M,1)=A(M,1)+(NT*ER)
4130 PRINT"■"NT"KG EXCHANGED FOR $
    "NT*ER:FORF=1TO2000:NEXT:RETURN
```

```
4000 PRINT"■■"
4020 PRINT"■■■□□□EXCHANGE
    AGENCY□□□□"
4030 PRINT"■■THE CURRENT
    EXCHANGE□□ RATE IS:—"
4040 PRINT:PRINT"1 KG=$"ER
4050 PRINT:PRINT"■ENTER NO.OF KG
    TO□□□□EXCHANGE":INPUTNT
4060 IFNT>A(M,3)THENPRINT"■YOU DO
    NOT HAVE THAT AMOUNT OF
    GOLD□!□!□!□□"
4070 NT=INT(NT)
4090 IFNTE>A(M,3)ORNTE<0THEN4020
4100 A(M,3)=A(M,3)-NT:A(M,2)=A(M,2)
    +(NT*ER):A(M,1)=A(M,1)+(NT*ER)
4130 PRINT"■"NT"KG EXCHANGED FOR":
    PRINT"■$"NT*ER:FORF=1TO2000:
    NEXT:RETURN
```

```
4000 CLS
4020 PRINTTAB(12,3)"EXCHANGE
    AGENCY"TAB(5,10) "THE CURRENT
    EXCHANGE RATE IS:—"TAB(5,12) "1 kg
```



```
    OF GOLD□ = □$";ER;:INPUT '''"ENTER
    NO. OF kg TO EXCHANGE",NTE
4Ø7Ø NTE=INT(NTE)
4Ø8Ø IF NTE>A(M,3) THEN PRINT'''"YOU
    DON'T HAVE THAT MUCH GOLD!"
4Ø9Ø IF NTE>A(M,3)OR NTE<Ø THEN
    PRINTTAB(28,14)SPC(1Ø):GOTO 4Ø2Ø
4Ø95 VDU11:PRINTSPC(39)
41ØØ A(M,3)=A(M,3)-NTE:A(M,2)=A(M,2)
    +(NTE*ER):A(M,1)=A(M,1)+(NTE*ER)
413Ø PRINT'';NTE"kg EXCHANGED□FOR□$";
    NTE*ER;SPC(2Ø):Z=INKEY(34Ø): RETURN
5ØØØ RETURN
```

```
4ØØØ CLS
4Ø2Ø PRINT@7,"exchange agency":
    PRINT@128,"THE CURRENT EXCHANGE
    RATE IS:—":PRINT@197,"1KG OF
    GOLD=";ER:PRINT@288,"ENTER NO. OF
    KG TO EXCHANGE";:INPUT NT
4Ø8Ø NT=INT(NT)
4Ø9Ø IF NT>A(M,2) OR NT<Ø THEN 4Ø2Ø
41ØØ A(M,2)=A(M,2)-NT:A(M,1)=A(M,1)
    +(NT*ER):A(M,Ø)=A(M,Ø)+(NT*ER)
413Ø PRINT@448,NT;"KG EXCHANGED FOR";
    NT*ER:FORZ=1TO1ØØØ:NEXT:RETURN
5ØØØ RETURN
```

Line 4ØØØ sets up the program.

Line 4Ø2Ø PRINTs the title of the screen, the current exchange rate, and prompts for the number of kilograms to be exchanged—the Commodores use Lines 4Ø2Ø to 4Ø5Ø. In all the programs except the Dragon/Tandy, Line 4Ø7Ø checks if there is sufficient gold held. Line 4Ø8Ø ensures the amount of gold exchanged is a whole number.

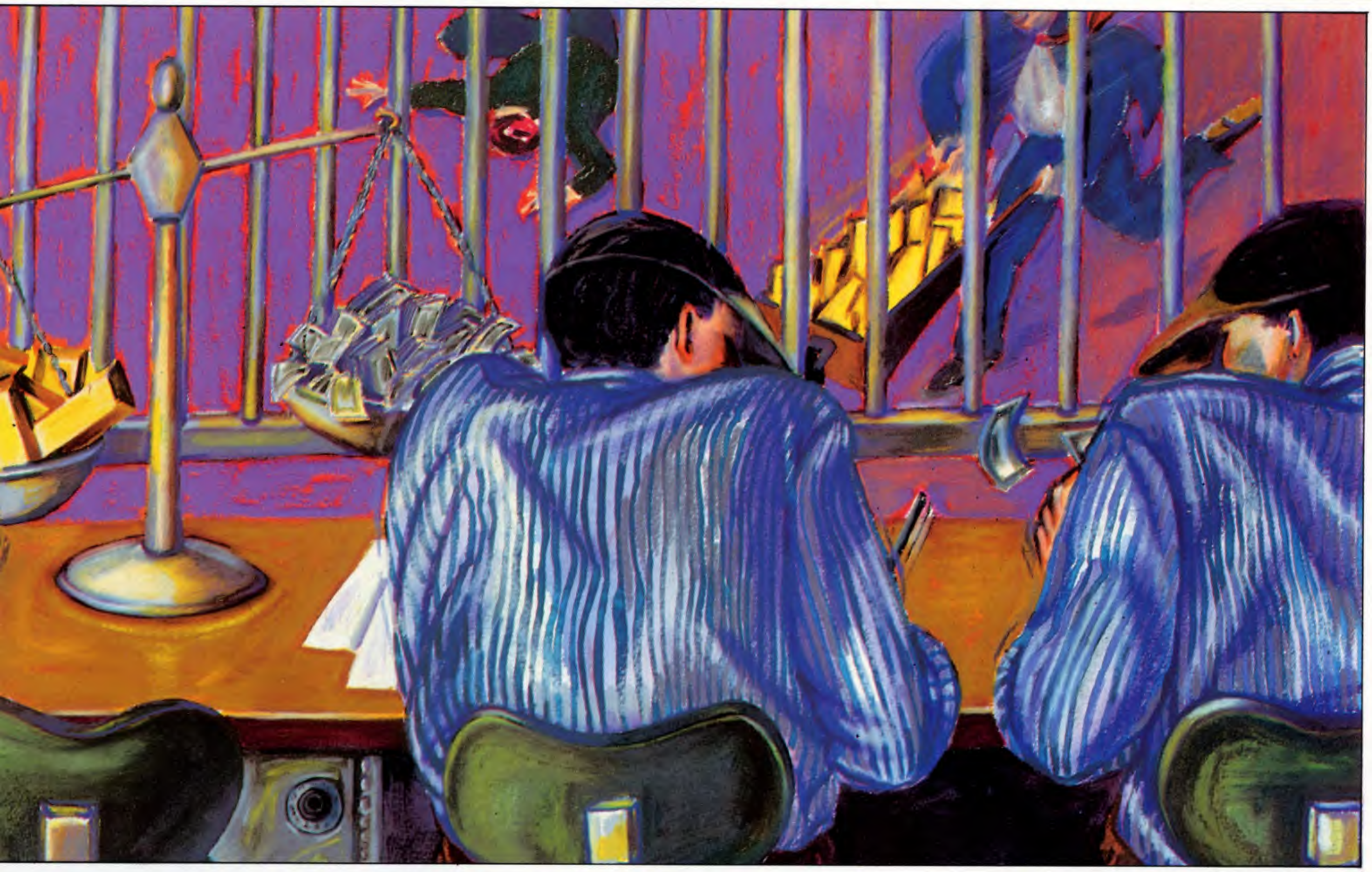
Line 4Ø9Ø sends the program back to the prompt if the amount exchanged is more than that being held, or less than zero. Line 41ØØ adjusts the assets according to the amount of gold that's been exchanged.

The subroutine tells the player how much gold has been exchanged for what amount in dollars, in Line 413Ø. Line 5ØØØ in the Spectrum and Acorn programs is the pass option.

FINISHING TOUCHES

```
1 FOR n=USR"a" TO USR "o"+7:
    READ a: POKE n,a: NEXT n
7ØØØ PAPER 5: INK Ø: BORDER 5: CLS
7Ø1Ø PRINT AT 9,12;a$(m): PRINT AT
    1Ø,8;"has gone bust!": PRINT FLASH 1;AT
    2Ø,1; "□Press any key to play again□"
7Ø3Ø PAUSE Ø: RUN 5
8ØØØ DATA 255,85,17Ø,Ø,Ø,Ø,Ø,Ø,62,28,
    56,126,28,62,12Ø,28
8Ø1Ø DATA 255,255,62,126,127,6Ø,124,
    126,Ø,Ø,Ø,Ø,1,1,1,1
8Ø2Ø DATA 7,29,49,45,255,255,91,126,
    128,96,48,8Ø,152,14Ø,252,138
8Ø3Ø DATA 1,1,1,49,49,49,49,255,122,
    187,62,95,153,255,153,126
8Ø4Ø DATA 2Ø9,177,224,128,128,128,128,
    128,Ø,Ø,128,128,64,32,32,16
8Ø5Ø DATA 1,3,7,7,4,4,7,7,255,255,255,
    255,149,149,159,159
8Ø6Ø DATA 24,126,153,255,126,153,126,
    219,128,192,224,24Ø,248,168,248,255
8Ø7Ø DATA 16,8,8,4,14,31,31,255
```

```
7ØØØ POKE5328Ø,3:POKE53281,3:
    PRINT"■■"
7Ø1Ø PRINT"■■■■■■■
    ■■"TAB(16);A$(M):PRINTTAB
    (12)"■■HAS GONE BUST!!!"
7Ø15 FORZ=1TO1ØØØ:NEXT
7Ø2Ø PRINTTAB(6);"■■■PRESS ANY
    KEY TO PLAY AGAIN"
7Ø3Ø POKE198,Ø:WAIT198,1:RUN5
6ØØØØ POKE56334,Ø:POKE1,35
```



```
60010 FORF=0TO2047:POKE12288+
    F,PEEK(53248+F):NEXT
60020 POKE1,39:POKE56334,1
60030 SP=12808
60040 READA:IFA=-1THEN60070
60050 POKESP,A
60060 SP=SP+1:GOTO60040
60070 RETURN
60080 DATA255,85,170,0,0,0,0,0,
    62,28,56,126,28,62,120,28
60090 DATA255,255,62,126,127,60,
    124,126,0,0,0,0,1,1,1,1
60100 DATA7,29,49,45,255,255,91,
    126,128,96,48,80,152,140,252,138
60110 DATA1,1,1,49,49,49,49,255,
    122,187,62,95,153,255,153,126
60120 DATA209,177,224,128,128,128,
    128,128,0,0,128,128,64,32,32,16
60130 DATA1,3,7,7,4,4,7,7,255,255,
    255,255,149,149,159,159
60140 DATA24,126,153,255,126,153,
    126,219,128,192,224,240,248,168,
      248,255
60150 DATA16,8,8,4,14,31,31,255,-1
```

```
7000 PRINT"  "
7010 PRINT"       
    "TAB(6);A$(M):PRINT"  
    HAS GONE BUST!!!"
7015 FOR Z=1TO1000:NEXT
7020 PRINT"    PRESS ANY KEY TO
    PLAY"
7030 POKE198,0:WAIT198,1:RUN
```

```
7000 CLS
7010 PRINTTAB(10,10)A$(M)" HAS
    GONE BUST"TAB(0,29)"PRESS ANY
    KEY TO PLAY AGAIN"
7030 Z=GET:RUN
8000 DATA 0,0,0,0,1,1,1,1
8010 DATA 7,29,49,45,255,255,91,126,
    128,96,48,80,152,140,252,138
8020 DATA 1,1,1,49,49,49,49,255,122,
    187,62,95,153,255,153,126
8030 DATA 209,177,224,128,128,128,128,
    128,0,0,128,128,64,32,32,16
8040 DATA 1,3,7,7,4,4,7,7,255,255,255,
    255,149,149,159,159
8050 DATA 24,126,153,255,126,153,126,
    219,128,192,224,240,248,168,248,255
8060 DATA 16,8,8,4,14,31,31,255,255,85,
    170,0,0,0,0,0
8070 DATA 62,28,56,126,28,62,120,28,
    255,255,62,126,127,60,124,126
```

```
7000 CLS
7010 PRINT@76,A$(M):PRINT@168,"HAS
    GONE BUST !":PRINT@449,"PRESS ANY
    KEY TO PLAY AGAIN"
7020 IF INKEY$="" THEN 7020 ELSE RUN
9000 FOR K=1 TO LEN(Z$)
9010 B$=MID$(Z$,K,1)
9020 IF B$>="0" AND B$<="9" THEN
    DRAW NU$(VAL(B$)):GOTO 9050
9030 IF B$="-" THEN DRAW"BF2R4"
9050 NEXT
9060 RETURN
```

Lines 7000 to 7030 are an 'another go?' routine.

In the Spectrum and Acorn programs, Lines 8000 to 8070 are the DATA for the UDGs. The Commodore 64 program uses Lines 60000 to 60150 to POKE the DATA for the UDGs. The Acorn program has the UDG DATA from Lines 8000 to 8070. The Dragon/Tandy program uses Lines 9000 to 9060 to draw the depth along the left hand side of the mining display screen.

Now you can amass your vast fortune and buy those things you always promised yourself!

COMMODORE HI-RES GRAPHICS

Now that you've got your Commodore into @HIRES mode try adding these commands to extend its limited graphics capabilities

In the first part of this article (see pages 748 to 751), you saw how to set up the main routine for a machine code program that would add graphics commands to the Commodore 64's BASIC. At that time, only two of the graphics commands were covered, though—@HIRES and @COLOUR.

In this part, find out how to add the subroutines which handle nine more graphics commands. And in part three, a further eleven will be covered.

PARAMETERS

Of course, these new instructions that you are adding to your Commodore's BASIC don't only work with the programs given in *INPUT*. You can use them in your own programs as well. But to do that you have to know about the parameters which follow the commands.

In the first part of this article the commands @HIRES and @COLOUR were introduced. @HIRES moves your Commodore into high resolution mode and needs to be followed by two parameters. The first specifies the colour being used to plot with. And the second—which is separated from the first by a comma—specifies the background colour.

@COLOUR is also followed by two parameters. The first figure you give specifies the border colour. And the second—again, separated from the first by a comma—specifies the background colour of the low-resolution screen. The background colour of the high-resolution screen is specified by the @HIRES command's parameters of course.

The figures used for the parameters correspond to the colours given in the Commodore 64's Programmers' Reference Guide.

RETURN TO @NRM

The @NRM command returns your Commodore from a multi-coloured or high-resolution graphics screen to the normal low-resolution screen.

```
ORG  5Ø912
JSR  $ØØ73
LDA  #$93
NOP
NOP
NOP
LDA  #$9B
STA  $DØ11
LDA  #$15
STA  $DØ18
LDA  #$C8
STA  $DØ16
JMP  $CA47
```

As this command simply undoes what several of the other graphics commands do, it does not require any parameters.

CHARACTER SET

The @CSET command allows you to choose the character set you want to use. When @CSET is followed by Ø, it selects the capital letters and graphics character set. And @CSET 1 selects the upper and lower case letters set.

@CSET followed by a 2 recalls the last graphics screen that was shown. Be careful with this command though. If the last graphics screen displayed used the multi-coloured mode, you have to follow @CSET 2 with the command @MULTI with the same parameters that were used before.

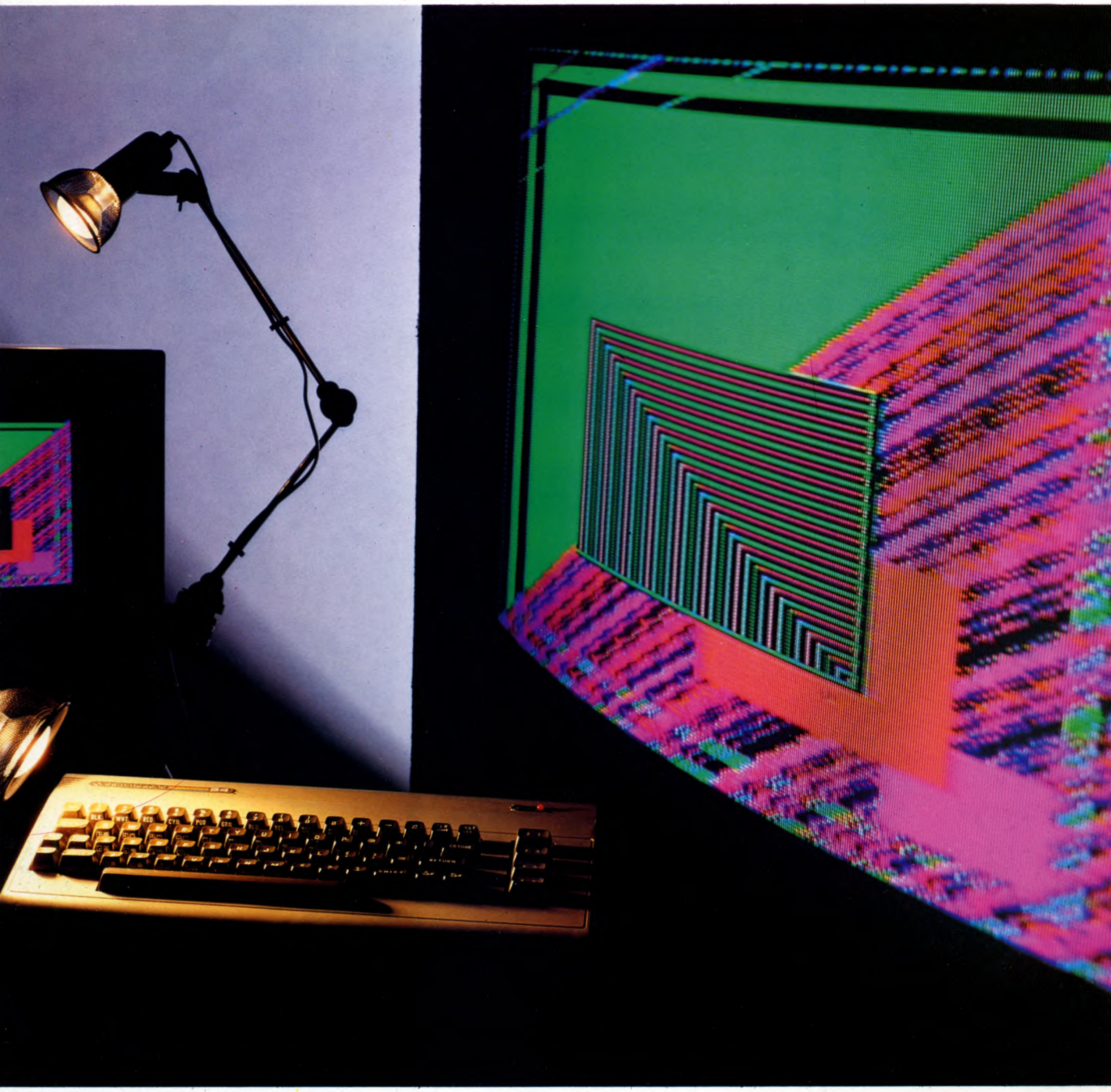
```
ORG  49568
JSR  $B79B
TXA
CMP  #$ØØ
BEQ  $C1B5
CMP  #$Ø1
BEQ  $C1BD
CMP  #$Ø2
BEQ  $C1C5
LDX  #$ØB
JMP  $ØØ79
JMP  $CA79
CLC
BNE  $C2Ø7
ADC  SA9C9,Y
BYT  $17
STA  $DØ18
JMP  $C1E9
LDA  #$3B
STA  $DØ11
LDA  $DØ18
AND  #$FØ
ORA  #$Ø8
STA  $DØ18
LDX  #$ØØ
LDA  $Ø2
STA  $Ø4ØØ,X
STA  $Ø4FA,X
STA  $Ø5F4,X
STA  $Ø6EE,X
INX
CPX  #$FB
BNE  $C1D8
LDX  $CØØE
JMP  $ØØ79
```

MULTI-COLOURED MODE

The command @MULTI usually follows the @HIRES command and multiplies @HIRE's one plotting colour into three. So @MULTI takes three parameters separated by commas, each of which specifies one of the colours. These colours are then referred to by the @PLOT command.

The following routine handles the @MULTI command:





```
ORG  49648
JSR  $B79B
TXA
STA  $CFFØ
STA  $DØ21
JSR  $AEFD
JSR  $B79E
TXA
STA  $CFF1
STA  $DØ22
JSR  $AEFD
JSR  $B79E
TXA
STA  $CFF2
STA  $DØ23
LDA  #$D8
STA  $DØ16
LDX  $CØØE
JMP  $ØØ79
```

CHANGING COLOUR

The command @LOWCOL allows you to change the colour used for plotting from that originally specified with @HIRES or @MULTI. It takes three parameters which, again, specify the three plotting colours to be used. Although high resolution mode only uses two colours—one for plotting and one background—three numbers must still follow the command, even though the third parameter has no effect. Note there is no space between LOW and COL. The following routine deals with @LOWCOL.

```
ORG  46696
JSR  $B79B
TXA
STA  $DØ21
JSR  $AEFD
JSR  $B79E
TXA
STA  $DØ22
JSR  $AEFD
JSR  $B79E
TXA
STA  $DØ23
LDA  #$8Ø
STA  $Ø2
LDX  $CØØE
JMP  $ØØ79
```

CHANGING COLOURS BACK

If you want to change your plotting colours back to their original values, you use the @HICOL command. And as it simply undoes the effects of the @LOWCOL commands, it needs no parameters. Note there is no space between HI and COL.

The following routine puts it into effect:

```
ORG  51712
JSR  $ØØ73
LDA  #$FF
STA  $CFF6
STA  $CFF7
STA  $CFF8
STA  $CFF9
LDX  $CØØE
JMP  $ØØ79
```

PLOTTING

Although @PLOT is one of the simplest commands, it is much longer than the other routines here. That's because it is a crucial part of the program, which the other routines call.

All @PLOT does is draw a single dot on the screen. It is followed by three parameters.

The first specifies the horizontal position of the dot. The second specifies its vertical position. And the third fixes how the dot is to be printed.

In both @HIRES and @MULTI mode, a Ø in the third parameter clears the dot by overprinting it with the background colour. A 1 in @HIRES mode draws a dot on the screen in the plotting colour specified by the @HIRES command. And a 2 reverses the dot—if it is in the plotting colour it changes it to the background colour, and if it is in the background colour it changes it to the plotting colour.

In @MULTI mode, if the third figure is a 1, 2 or 3, the dot is drawn in the first, second or third colour specified by the @MULTI or @LOW COL command. This time, 4 reverses the colour—it changes a dot in the background colour to one in the third colour, a dot in the first colour to one in the second, a dot in the second colour to one in the first, and one in the third colour back to one in the background colour.

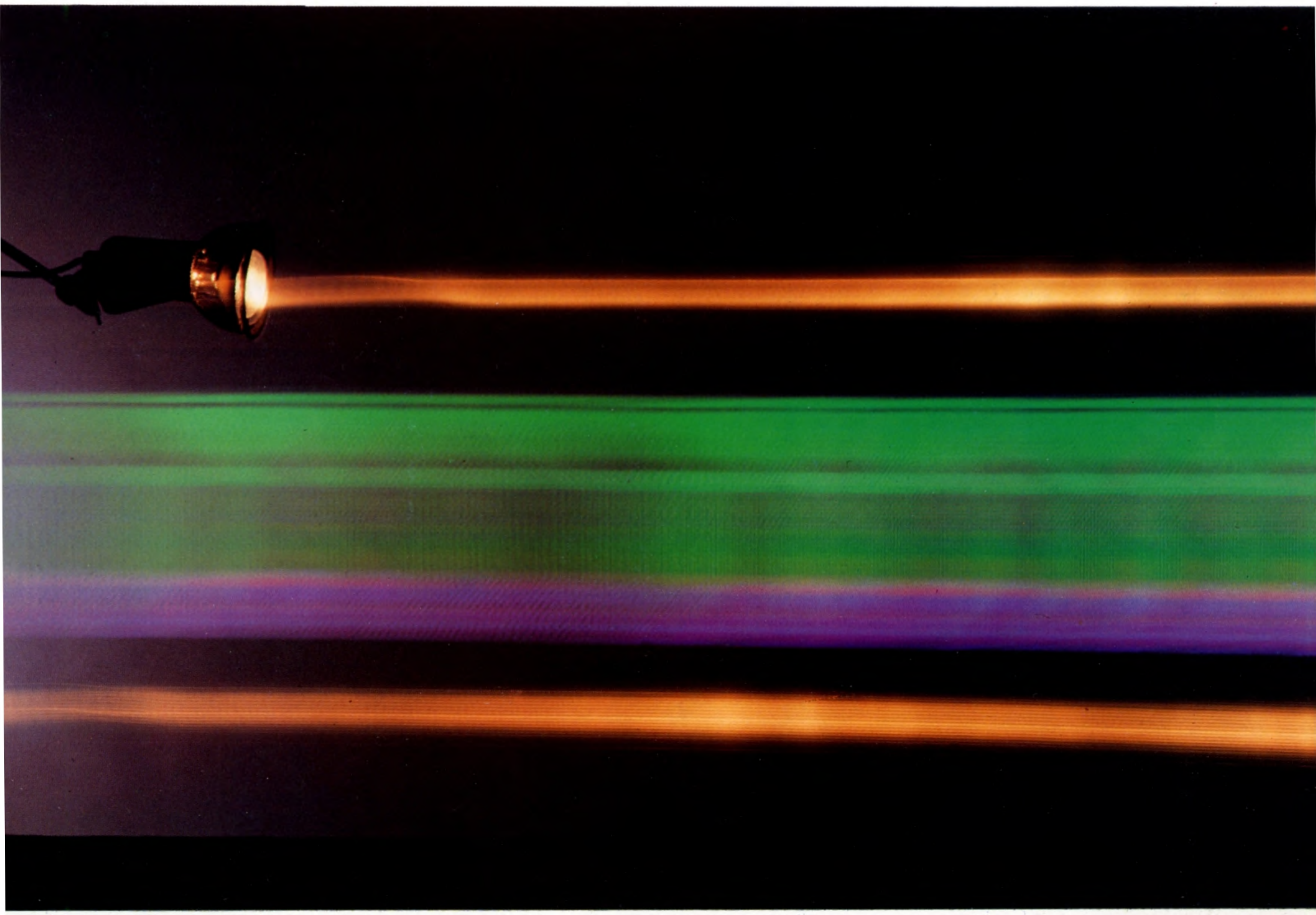
Drawing one dot on the screen may seem like a pretty minor facility. But it is the foundation of many of the other commands. Obviously, when you draw a line on the screen it is made up of a number of dots placed repeatedly, next to each other. And once you can create lines, you can extend the process to make circles, rectangles and block commands.

Anyway, the following routine supplies the building blocks, the dots themselves:

```
ORG  49811
LDA  #$ØØ
STA  $CFFF
JSR  $ØØ73
JSR  $AD8A
JSR  $B7F7
LDA  $14
STA  $CFEØ
LDA  $15
STA  $CFE1
JSR  $AEFD
JSR  $B79E
STX  $CFE2
JSR  $AEFD
JSR  $B79E
TXA
AND  #$Ø7
STA  $CFE3
LDA  $CFE2
CMP  #$C9
BCC  $C2CA
JMP  $C48C
LDA  $CFEØ
CMP  #$4Ø
BCC  $C2D9
LDA  $CFE1
BEQ  $C2D9
JMP  $C48C
JMP  $C36E
LDA  $CFE2
STA  $CFE8
STA  $CFE5
LDA  $CFEØ
STA  $CFE9
```



```
STA   $CFE6
LDA   $CFE1
STA   $CFEA
STA   $CFE7
LDA   $CFE8
LSR   A
LSR   A
LSR   A
STA   $CFEB
LDA   $CFE9
STA   $CFEC
LDA   $CFEA
LSR   A
ROR   $CFEC
LSR   A
ROR   $CFEC
LSR   A
ROR   $CFEC
STA   $CFED
LDA   $CFE8
AND   #$07
STA   $CFEE
LDA   $CFE9
AND   #$07
STA   $CFEF
LDA   #$07
SEC
SBC   $CFEF
STA   $CFEF
LDA   #$00
STA   $FB
LDA   #$20
STA   $FC
LDX   $CFEB
BEQ   $C34E
INC   $FC
LDA   $FB
CLC
ADC   #$40
STA   $FB
BCC   $C34B
INC   $FC
DEX
BNE   $C33E
LDA   $CFEC
ASL   A
ASL   A
ASL   A
BCC   $C359
INC   $FC
CLC
ADC   $FB
STA   $FB
BCC   $C361
INC   $FC
LDA   $FB
CLC
ADC   $CFEE
STA   $FB
BCC   $C36D
INC   $FC
RTS
JSR   $C2DC
LDA   $D016
AND   #$10
CMP   #$10
BEQ   $C397
LDA   $CFE3
CMP   #$00
BNE   $C387
JSR   $C491
JMP   $C3F9
CMP   #$02
BNE   $C391
JSR   $C4AB
JMP   $C3F9
JSR   $C49E
JMP   $C3F9
CLC
LDA   $CFE0
CMP   #$A0
BCC   $C3A2
JMP   $C48C
LDA   $CFE1
CMP   #$00
BEQ   $C3AC
JMP   $C48C
LDA   $CFE0
ASL   A
STA   $CFE0
LDA   $CFE1
ADC   #$00
STA   $CFE1
JSR   $C2DC
LDA   $CFEF
LSR   A
STA   $CFEF
LDA   $CFE3
CMP   #$00
BNE   $C3D2
JSR   $C4B8
JMP   $C3F9
CMP   #$04
BNE   $C3DC
JSR   $C4C5
JMP   $C3F9
CMP   #$01
BNE   $C3E9
NOP
NOP
NOP
JSR   $C4D2
JMP   $C3F9
CMP   #$02
BNE   $C3F6
NOP
NOP
NOP
JSR   $C4DF
JMP   $C3F9
JSR   $C4EC
CLC
LDA   $CFE1
AND   #$01
BEQ   $C40C
LDA   $CFE0
LSR   A
ADC   #$80
LSR   A
LSR   A
JMP   $C412
LDA   $CFE0
LSR   A
LSR   A
LSR   A
STA   $FB
LDA   $CFE2
LSR   A
LSR   A
LSR   A
STA   $FC
LDY   #$00
STY   $FD
LDX   #$04
```



```
STX $FE
CPY $FC
BEQ $C438
LDA $FD
ADC #$28
STA $FD
LDA $FE
ADC #$00
STA $FE
INY
JMP $C424
CLC
LDA $FD
ADC $FB
STA $FD
LDA $FE
ADC #$00
STA $FE
LDY #$00
LDA $CFF9
CMP #$FF
BNE $C46E
LDA $D016
AND #$10
CMP #$10
BEQ $C45A
JMP $C480
LDA $CFF3
STA ($FD),Y
CLC
LDA $FE
ADC #$D4
STA $FE
LDA $CFF2
STA ($FD),Y
JMP $C480
NOP
LDA $CFF9
STA ($FD),Y
CLC
LDA $FE
ADC #$D4
STA $FE
LDA $CFF8
STA ($FD),Y
LDA $CFFF
BEQ $C486
RTS
LDX $C00E
JMP $0079
LDX #$0B
JMP ($0300)
LDY #$00
LDX $CFEF
LDA ($FB),Y
AND $C120,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
ORA $C118,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
EOR $C118,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
AND $C0AA,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
EOR $C0AE,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
ORA $C0B2,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
ORA $C0B6,X
STA ($FB),Y
RTS
LDY #$00
LDX $CFEF
LDA ($FB),Y
ORA $C0BA,X
STA ($FB),Y
RTS
```

DRAWING A LINE

Again, aggregating dots to make up lines seems like a simple enough task. But the line routine is going to be called by other routines—circle, rectangle, block—which make up their shapes from lines.

@LINE takes five parameters. The first two specify the horizontal and vertical positions of the beginning of the line. The second pair specify the horizontal and vertical positions of the end of the line. And the fifth parameter specifies the colour of the line in exactly the same way as the third parameter of @PLOT did.

The following routines draws lines:

```
ORG 50045
LDA #$00
STA $CFDE
STA $CFD3
STA $CFD5
JSR $0073
JSR $AD8A
JSR $B7F7
LDA $14
STA $CFE0
LDA $15
STA $CFE1
JSR $AEFD
JSR $B79E
STX $CFE2
JSR $AEFD
JSR $AD8A
JSR $B7F7
LDA $14
STA $CFD0
LDA $15
STA $CFD1
JSR $AEFD
JSR $B79E
STX $CFD2
JSR $AEFD
JSR $B79E
STX $CFE3
LDA $CFE2
CMP #$C9
BCC $C3D3
JMP $C54F
LDA $CFD2
CMP #$C9
BCC $C3DD
JMP $C54F
LDA $CFE3
AND #$07
STA $CFE3
JSR $C2AE
LDA $CFD0
SEC
SBC $CFE0
STA $CFD7
LDA $CFD1
SBC $CFE1
STA $CFD8
BPL $C411
DEC $CFD5
SEC
LDA #$00
SBC $CFD7
STA $CFD7
LDA #$00
SBC $CFD8
STA $CFD8
LDA #$00
STA $CFD6
LDA $CFD2
SEC
SBC $CFE2
STA $CFD9
LDA $CFD3
SBC $CFDE
STA $CFDA
BPL $C43F
DEC $CFD6
SEC
LDA #$00
SBC $CFD9
STA $CFD9
LDA #$00
SBC $CFDA
STA $CFDA
LDA #$00
STA $CFDD
LDA $CFD9
SEC
SBC $CFD7
LDA $CFDA
SBC $CFD8
BCC $C46E
LDX $CFD9
LDA $CFD7
STA $CFD9
STX $CFD7
LDX $CFDA
LDA $CFD8
STA $CFDA
STX $CFD8
DEC $CFDD
LDA $CFDD
STA $CFDB
LDA $CFD8
STA $CFDC
JSR $C2AE
LDA $CFDD
BNE $C494
LDA $CFE0
CMP $CFD0
BNE $C4A7
LDA $CFE1
CMP $CFD1
BNE $C4A7
BEQ $C4A4
LDA $FE2
CMP $CDE2
BNE $C4A7
LDA $CFDE
CMP $CFD3
BNE $C4A7
JMP $C538
LDA $CFDD
BNE $C4B2
JSR $C4FF
JMP $C4B5
JSR $C519
JSR $C4D7
JSR $C4D7
BPL $C4D1
LDA $CFDD
BNE $C4C8
JSR $C519
JMP $C4CB
JSR $C4FF
JSR $C4EB
JSR $C4EB
JSR $C2AE
JMP $C47D
LDA $CFDB
SEC
SBC $CFD9
STA $CFDB
LDA $CFDC
SBC $CFDA
STA $CFDC
RTS
LDA $CFDB
CLC
ADC $CFD7
STA $CFDB
LDA $CFDC
ADC $CFD8
STA $CFDC
RTS
LDA $CFD5
BNE $C50D
INC $CFE0
BNE $C50C
INC $CFE1
RTS
LDA $CFE0
BNE $C515
DEC $CFE1
DEC $CFE0
RTS
LDA $CFD6
BNE $C527
INC $CFE2
BNE $C526
INC $CFDE
RTS
LDA $CFE2
BNE $C52F
DEC $CFDE
DEC $CFE2
RTS
LDX #$0B
JMP ($0300)
LDA $CFFE
CMP #$00
BEQ $C540
RTS
LDX $C00E
JMP $0079
LDX $AA00,Y
```

DRAWING RECTANGLES

@REC draws a rectangle and requires five parameters. The first pair specify the horizontal and vertical co-ordinates of the top lefthand corner of the rectangle. The third parameter specifies the width of the rectangle, and the fourth parameter its depth. Again, the fifth specifies the colour in the same way as the third parameter of @PLOT. The following program draws a rectangle:

```
ORG 50944
LDA #$02
STA $CFFF
LDA #$00
STA $CFDE
STA $CFD3
STA $CFD5
JSR $0073
JSR $AD8A
JSR $B7F7
LDA $14
STA $C841
LDA $15
STA $C842
JSR $AEFD
JSR $B79E
STX $C843
JSR $AEFD
JSR $AD8A
JSR $B7F7
LDA $14
STA $C844
LDA $15
STA $C845
JSR $AEFD
JSR $B79E
STX $C846
JSR $AEFD
JSR $B79E
STX $C847
LDA $C843
CMP #$C9
```

```
BCS  $C79D
LDA  $C842
CMP  #$01
BNE  $C766
LDA  $C841
CMP  #$3F
BPL  $C79D
LDA  $C841
CLC
ADC  $C844
STA  $C848
LDA  $C842
ADC  #$00
ADC  $C845
STA  $C849
CMP  #$00
BEQ  $C78A
CMP  #$01
BPL  $C79D
LDA  $C848
CMP  #$63
BPL  $C79D
LDA  $C843
CLC
ADC  $C846
BCS  $C79D
CMP  #$C9
BCS  $C79D
STA  $C84A
JMP  $C7A2
LDX  #$0B
JMP  ($0300)
LDA  $C847
STA  $CFE3
LDA  $C841
STA  $CFE0
LDA  $C842
STA  $CFE1
LDA  $C843
STA  $CFE2
STA  $CFD2
LDA  $C848
STA  $CFD0
```

```
LDA  $C849
STA  $CFD1
JSR  $C56D
LDA  $C848
STA  $CFE0
STA  $CFD0
LDA  $C849
STA  $CFE1
STA  $CFD1
LDA  $C84A
STA  $CFD2
LDA  $C843
STA  $CFE2
JSR  $C56D
LDA  $C84A
STA  $CFE2
STA  $CFD2
LDA  $C841
STA  $CFE0
LDA  $C842
STA  $CFE1
LDA  $C848
STA  $CFD0
LDA  $C849
STA  $CFD1
LDA  $C84A
STA  $CFE2
STA  $CFD2
JSR  $C56D
LDA  $C843
STA  $CFE2
LDA  $C84A
STA  $CFD2
LDA  $C841
STA  $CFE0
STA  $CFD0
LDA  $C842
STA  $CFE1
STA  $CFD1
JSR  $C56D
LDX  $C00E
JMP  $0079
```

FILLING IN A BLOCK

@BLOCK draws a solid rectangle—in other words, it draws a rectangle like the one in the routine above, then fills it with colour. Its five parameters work the same as those for @REC.

The following routine creates solid rectangular blocks of colour:

```
ORG  51328
LDA  #$02
STA  $CFFF
LDA  #$00
STA  $CFDE
STA  $CFD3
STA  $CFD5
JSR  $0073
JSR  $AD8A
```

```
JSR  $B7F7
LDA  $14
STA  $C94E
LDA  $15
STA  $C94F
JSR  $AEFD
JSR  $B79E
STX  $C950
JSR  $AEFD
```

```
JSR  $AD8A
JSR  $B7F7
LDA  $14
STA  $C955
LDA  $15
STA  $C956
JSR  $AEFD
JSR  $B79E
STX  $C957
JSR  $AEFD
JSR  $B79E
STX  $C954
LDA  $C950
CMP  #$C9
BCS  $C8FE
LDA  $C94F
CMP  #$01
BNE  $C8E6
LDA  $C94E
CMP  #$3F
BPL  $C8FE
LDA  $C956
CMP  #$01
BNE  $C8F4
LDA  $C955
CMP  #$3F
BPL  $C8FE
LDA  $C957
CMP  #$C9
BCS  $C8FE
LDX  #$00
```

```
LDA  $C94E
STA  $CFE0
LDA  $C94F
STA  $CFE1
LDA  $C955
STA  $CFD0
LDA  $C956
STA  $CFD1
LDA  $C958
STA  $CFD2
STA  $CFE2
CMP  $C957
BEQ  $C948
CLC
ADC  #$01
STA  $C958
JSR  $C56D
JMP  $C916
LDX  $C00E
JMP  $0079
JMP  $C903
LDX  #$0B
JMP  ($0300)
LDA  $C957
SEC
SBC  $C950
BEQ  $C8FE
BCC  $C8FE
LDA  $C950
STA  $C958
```

THE ROUTINE TABLE

Don't forget to update the table that contains the start addresses of the command routines to include the new ones you have just added. The routine table should now read:

```
ORG  49345
WOR  &C130
WOR  &C100
WOR  &C1F0
WOR  &C6E0
WOR  &C220
WOR  &CA00
WOR  &C293
WOR  &C370
WOR  &C860
WOR  &CE00
WOR  &CE00
WOR  &C1A0
```

```
WOR  &C700
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
WOR  &CE00
```

TESTING THE COMMANDS

Try the following BASIC graphics program to test the commands you have added:

```
1 PRINT "[graphic][graphic]";TAB(9);"THIS IS A DEMO
   PROGRAM"
2 FOR Z=1 TO 10:@CSET(0):
   FOR ZZ=1 TO 100:NEXT ZZ
3 @CSET(1):FOR ZZ=1 TO 100:
   NEXT ZZ,Z
10 @HIRES 0,3:@COLOUR 0,0:
   @MULTI 2,4,5
20 ZZ=0:FOR Z=1 TO 199:ZZ=ZZ+
   .8:@LINE 160-ZZ,199,160,199-Z,
   RND(1)*3+1:NEXT Z
30 FOR Z=1 TO 3:@BLOCK 160-
   Z*40,199-Z*40,160-Z*15,199-
   Z*15,Z:NEXT Z
40 FOR Z=5 TO 15 STEP 5:@BLOCK
   Z,Z,160-Z,199-Z,4:NEXT Z
50 FOR Z=1 TO 75 STEP 2:@REC
   114-Z,154-Z,Z,Z,Z AND 3:NEXT Z
60 ZZ=0:FOR Z=1 TO 199 STEP
   2:ZZ=ZZ+1.6:@PLOT ZZ,0,3:@PLOT
   0,Z,3:NEXT Z
70 @LOWCOL 3,6,7:FOR Z=1 TO 3
80 @LINE Z*30,0,0,Z*15,Z:
   NEXT Z
90 FOR Z=1 TO 4000:NEXT Z
100 FOR Z=1 TO 10:@CSET(0):
   FOR ZZ=1 TO 100:NEXT ZZ:
   @CSET(2):@MULTI 2,4,5
105 FOR ZZ=1 TO 100:NEXT ZZ,Z
110 FOR Z=1 TO 2000:NEXT Z:
   @NRM:@COLOUR 6,12:PRINT "[graphic]"
```

Switch on the routine with SYS 49152.

THE LOW-RES SCREEN

When you @NRM or @CSET 0, you return to the low-res screen. This routine does that:

```
ORG  51744
LDX  #$00
LDA  $0400,X
STA  $A028,X
LDA  $0500,X
STA  $A128,X
LDA  $0600,X
STA  $A228,X
LDA  $0700,X
STA  $A328,X
INX
BEQ  $CA40
JMP  $CA22
LDA  $02
LDY  #$00
JMP  $C174
LDA  $01
AND  #$FE
STA  $01
LDX  #$00
LDA  $A028,X
STA  $0400,X
LDA  $A128,X
```

```
STA  $0500,X
LDA  $A228,X
STA  $0600,X
LDA  $A328,X
STA  $06B0,X
INX
BEQ  $CA6D
JMP  $CA4F
LDA  $01
ORA  #$01
STA  $01
LDX  $C00E
JMP  $0079
LDA  $D011
AND  #$20
CMP  #$20
BNE  $CA85
JMP  $C979
LDX  $C00E
LDA  #$15
STA  $D018
JMP  $0079
```

You must also alter the @HIRES routine in part one by doing the following POKEs—POKE 49520,76: POKE 49521,23: POKE 49522,202. ReSAVE @HIRES after you have made these changes.

A PLAY ON WORDS

Additions, amendments and deletions, all of these are possible. So if you want to create the perfect composition, start with a draft and improve it at your leisure



In the second part of this article, the main core of the text editor, we describe how to use the editing functions on each machine. Although the general procedures for using the program follow much the same pattern, the specific controls and features do differ.

Once you have keyed in the part of the program contained in this article, you will be able to make full use of the editing facilities contained within the program. But remember, you won't be able to print out any text until you've keyed in the final part of the program which will be published in the third part of this article.

Since the art of letter writing has become a thing of the past, you will find the editing facilities contained in the text editor program helpful on many levels. At the simplest level, spelling corrections can be made just by moving your electronic nib—the cursor—to the offending word, and deleting or inserting characters as necessary.

If you don't have problems with spelling, but find actually composing the text a problem, particularly if it's an important letter such as an application for a job, or a letter to your bank manager explaining why you've just inadvertently overdrawn your account, then you'll find this program ideal. When typing a letter you can end up wasting many sheets of paper as you struggle to find the right words—the opening words are usually the most difficult to find. Composing straight onto your computer keyboard will keep the waste paper bin empty and your temper intact. Instead of agonizing over finding the perfect opener, you can simply do your letter in draft form. You can then analyze what you've written by scrolling the text—either backwards or forward—up and down the screen. Once you've decided what needs to be amended or omitted, you can delete or insert words, phrases or even sentences until you finally arrive at the perfect composition.

An additional benefit of composing onto the screen is that you can store the letter and use it again, or simply have a record of what you've said.

The facilities contained within this program are similar to, although considerably less sophisticated than, the facilities available on the modern phototypesetting machines which are used to typeset publications such as *INPUT*. And you can very easily see the advantages if you have read about, or are familiar with, the methods used to compose type before the phototypesetter was invented.

S

The Spectrum program is set up for use with a tape unit, so ensure you make the necessary amendments if you want to use a Microdrive.

The editing features are almost identical to those used for normal BASIC editing, and no special 'editor' mode is needed. All entries and amendments to copy have to be made in the bottom section of the screen displaying the work area—rather than the top section which is for viewing the text that is in the machine's memory.

When editing text that is in the work area, further characters can be inserted by using the CAPS SHIFT and 5 to move the cursor to the left and CAPS SHIFT and 8 to move it to the right. Once the cursor is in the required position, key in any additions. To delete, position the cursor to the right of the redund-

- MAKING ADDITIONS
- CORRECTING ERRORS
- MAKING AMENDMENTS
- MAKING DELETIONS
- ADJUSTING TEXT
- MOVING PARAGRAPHS
- CONVENIENCE OF INPUT
- TIME FOR ANALYSIS
- USING THE PROGRAM TO PERFECT COMPOSITION



ant character and press [CAPS SHIFT] and 0.

To put text from the work area into memory, use [ENTER]. Lines longer than 64 characters (the maximum two lines that can be held in the work area) are transferred automatically. But these remain linked to the following line for printout or display unless special formatting commands are used.

The cursor up and cursor down keys are used to locate specific lines of text contained in the memory—a bright marker is positioned *under* your 'target' line in the text viewing area towards the top of the screen. The line can be copied to the work area using the normal EDIT function—by pressing [SHIFT] and 1, then edited as described above. To delete a line of text already in the memory, press [CAPS SHIFT] and 9 to exit from editor mode, and press [CAPS SHIFT] and [SYMBOL SHIFT] simultaneously.

To scan text contained in the memory use cursor down and cursor up keys, [CAPS SHIFT] and 6 and [CAPS SHIFT] and 7. These keys will move the text up or down one line at a time.

```
1000 REM print screen
1005 PLOT 0,13: DRAW 255,0: PLOT 0,14:
   DRAW 255,0
1010 PRINT AT 0,0;: FOR n=p-10 TO
   p+8
1020 IF n<1 OR n>200 THEN PRINT s$:
   GOTO 1050
1025 IF n=p THEN PRINT
1030 PRINT t$(n)
1040 POKE 22528+320,120
1050 NEXT n
1060 RETURN
2000 REM input
2010 LET i$="": LET j$=""
2015 PRINT #1;AT 0,0;i$;FLASH 1; BRIGHT
   1;"□"; FLASH 0; BRIGHT 0;j$;"□"
2020 PAUSE 0: LET a$=INKEY$: IF a$=""
   THEN GOTO 2020
2025 BEEP .01,20
2030 IF a$<CHR$ 32 THEN GOSUB 2500
2040 IF a$>CHR$ 31 AND a$<CHR$ 123
   THEN LET i$=i$+a$
2042 IF a$=CHR$ 13 AND b=ext-6 THEN
   PRINT #1;AT 0,0;s$;s$; FLASH 1;"TEXT
   FILE FULL": BEEP 2,10: RETURN
2045 IF a$=CHR$ 13 OR LEN i$+LEN
   j$=64 THEN PRINT #1;AT 0,0;s$;s$;s$:
   LET i$=i$+j$: GOTO 2100
2050 IF a$=CHR$ 14 THEN RETURN
2052 IF a$=CHR$ 6 THEN INPUT "Enter
   target string", LINE z$: IF z$="" THEN
   GOTO 2052
2053 IF a$=CHR$ 6 THEN LET p=4:
   GOSUB 8000
2054 IF a$=CHR$ 4 THEN GOSUB 8000
2055 IF a$=CHR$ 5 THEN GOSUB 8500
2060 GOTO 2015
2100 IF LEN i$>32 THEN GOTO 2150
2105 FOR n=b+1 TO p STEP -1
2110 LET t$(n+1)=t$(n)
2120 NEXT n
2130 LET t$(n+1)=i$: LET p=p+1: LET
   b=b+1
2140 GOSUB 1000: GOSUB 2500: GOTO 2000
2150 FOR n=b+1 TO p STEP -1
2160 LET t$(n+2)=t$(n): LET
   t$(n+3)=t$(n+1)
2170 NEXT n
2180 LET t$(n+1)=i$(TO 32): LET
   t$(n+2)=i$(33 TO ): LET p=p+2: LET
   b=b+2
2190 GOTO 2140
2200 LET p=p-1: FOR n=p TO b+1
2210 LET t$(n)=t$(n+1)
2220 NEXT n
2225 LET b=b-1
2230 GOSUB 1000
2240 RETURN
2500 REM control codes
2520 IF a$=CHR$ 10 AND p<b-2 THEN
   LET p=p+1: GOSUB 1000
2530 IF a$=CHR$ 11 AND p>t+3 THEN
   LET p=p-1: GOSUB 1000
2540 IF A$=CHR$ 12 AND LEN i$>0 THEN
   LET i$=i$ (TO LEN i$-1)
2550 IF a$=CHR$ 8 AND LEN i$>0 THEN
   LET j$=i$(LEN i$)+j$: LET i$=i$ (TO
   LEN i$-1)
2560 IF a$=CHR$ 9 AND LEN j$>0 THEN
   LET i$=i$+j$(1): LET j$=j$(2 TO )
2570 IF a$<>CHR$ 7 THEN GOTO 2580
2572 LET j$=t$(p-1): LET i$="": PRINT
   #1;AT 0,0;s$;s$
2575 IF j$(LEN j$)=CHR$ 32 THEN LET
   j$=j$( TO LEN j$-1): IF LEN j$>0
   THEN GOTO 2575
2580 IF a$=CHR$ 15 AND p>4 THEN
   GOSUB 2200
2690 RETURN
3000 REM colours
3010 PRINT AT 10,4;"Select paper colour
   (0-7)"
3020 PAUSE 0: LET a$=INKEY$: IF a$<"0"
   OR a$>"7" THEN GOTO 3020
3030 PAPER VAL a$: BORDER VAL a$: CLS :
   RETURN
```

The procedures for using this program follow the general outlines given previously. The

standard Commodore 64 screen editing facilities are very effective and these are retained in this text editor. But when you enter edit mode, editing controls pass largely to the function keys, some of which are used by the extra features associated with the printer routine.

Text is created or amended in the work area near the bottom of the screen. The upper part of the display is for viewing text held in memory. Deletions and insertions are made in the normal way by using the [INST] [SHIFT] or [DEL] keys once you have positioned the cursor using the [CRSR] left and right keys. Existing characters are overwritten unless you use the insert mode which can be selected by pressing [f5].

Text is placed in memory and displayed in the viewing area by pressing [RETURN]. It may be recalled at any time by entering 'editor' mode which you do by pressing [f1]. A marker (■) indicates the text access point. The line immediately above the marker can be copied to the work area by pressing [f3], and there it may be edited in the normal way.

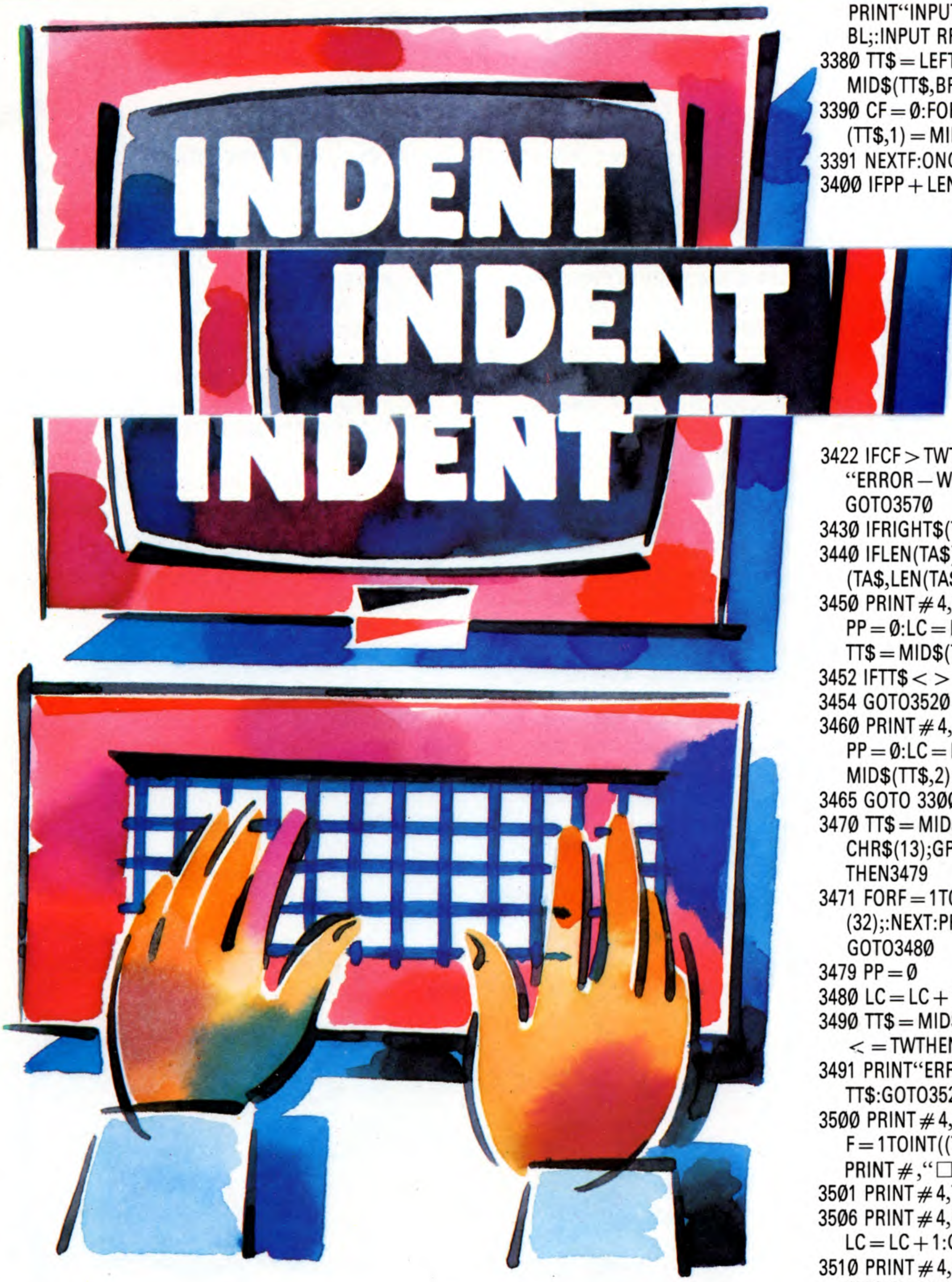
To delete a line of text in memory, enter editor mode and position the marker immediately above the line you wish to remove, then press [INST/DEL]. A line space may be entered by simultaneously pressing [INST] and [SHIFT], or by pressing [RETURN] in entry mode without entering text in the work area.

The contents of the memory can be viewed in edit mode by pressing [f1], followed by [CRSR] up or down with or without [SHIFT] to scroll up and down from the marker position to which the editor mode always returns.

You can jump five lines at a time by pressing the up arrow key or down by pressing the left arrow key.

To exit editor or entry modes (in sequence) press [f7]. Text in memory will remain unaffected.

```
1700 IFCP=1THEN1510
1710 CP=CP-1:FORF=CPTOTL:TX$(F)
   =TX$(F+1):NEXT:TL=TL-1
1720 GOSUB2090
1730 GOTO1510
1800 IFTL>499THEN1510
1810 FORF=TL+1TOCP+1STEP-1:TX$
   (F)
   =TX$(F-1):NEXT:TL=TL+1:TX$ (CP)
   =" "
1820 GOSUB2090:GOTO1510
2000 X=0:IF TL>499 THEN 2060
2001 IFLEN(A$)<41THENTT$(X)=LEFT$
   (A$,LEN(A$)-1):A$="":GOTO2030
2010 FORI=41TO1STEP-1:IFMID$(A$,I,
   1)<>"□"THENNEXT:I=41
2020 TT$(X)=LEFT$(A$,40):A$=MID$
   (A$,41)
2030 X=X+1:IFA$<>""ANDA$<>
   "□"THEN2001
2040 FOR I=TL+XTOCP+XSTEP-1:TX$(I)
   =TX$(I-X):NEXTI
2050 FORI=0TOX-1:TX$(CP+I)=TT$
   (I):NEXT
2060 A$="□":P=0:PRINTLEFT$
   (GC$,23)A$;
2080 TL=TL+X:CP=CP+X
2090 IF CP<15 THEN S1=0:GOTO 2100
2095 S1=CP-15
2100 PRINT"■";:FORK=S1TOS1+15:
   PRINTTX$(K);:IFLEN(TX$(K))<40
   THENPRINTCHR$(160)
2110 IFK=CP-1THENPRINT"■■□
   ■π"
2120 NEXT
2122 PRINT"■■"SP$"■0■123456
   789■0■123456789■0■123456
   789■0■123456789π■";
2130 PRINT"■MEM";BL$;"FREE=■";
   40*(501-TL);
2140 PRINT"■ O/W MODE=■"OW"
   ■ EDIT MODE=";EM:PRINT"■"
   SW$"π";:RETURN
2500 IFCP<5THENPM=CP
2510 GETTB$:IFTB$=""THEN2510
2512 IFPM=1ANDTB$="■"THEN2510
2520 IFPM>1ANDTB$="■"THENPM=
   PM-1:CP=PM:GOTO2550
2525 IFPM<TLANDTB$="■"THENPM=
   PM+1:CP=PM:GOTO2550
2530 IFPM>5ANDTB$="↑"THENPM=
   PM-5:CP=PM:GOTO2550
2535 IFPM<TL-5 ANDTB$="←"THEN
   PM=PM+5:CP=PM:GOTO2550
2540 GOTO2560
2550 GOSUB2090:GOTO2510
2560 IFTB$=CHR$(136)THENEM=0:
   GOSUB2090:GOTO1505
2570 IFTB$<>CHR$(148)THEN2580
2571 IFCP<1THEN2510
2572 FORK=TL+1 TO PM+1 STEP-1:
   TX$(K)=TX$(K-1):NEXT:
   TL=TL+1:TX$(PM)=""
2573 GOSUB 2090:GOTO2510
2580 IFTB$<>CHR$(20)THEN2590
2581 IFPM=TLTHEN2510
2582 FORK=PMTOTL:TX$(K)=TX$(K+1):
   NEXT:TL=TL-1
2583 TX$(TL+1)="":GOSUB2090:
   GOTO 2590
2584 CP=CP-1
2590 IFTB$="@"THENSF=SF+1:
   IFSF=1THENSS=CP:GOTO2510
2600 IFSF=2THENSE=CP:SF=0:
   GOSUB5130:GOTO2510
2610 IFTB$="S"THENGOSUB5070
2620 GOTO2510
3000 IFTL<2THEN3050
3010 PRINT"■■■"TAB(12)
   "■PRINTER ROUTINEπ":CLOSE4
3020 PRINT"■■■FROM ■M■EMORY
   OR FROM ■F■ILE ?"
3030 GETR$:IFR$<>"M"ANDR$<>"F"
   THEN3030
3040 IFR$="M"THEN3060
3050 GOSUB4500
3060 IFTL=1THENPRINTTAB(11)
   "■■■■■NO FILE IN
   MEMORYπ": GOTO3570
3070 KF=0:PRINT"■■FILL VARIABLE
   BLOCKS(Y/N)?"
3080 GETR$:IFR$<>"Y"ANDR$<>"N"
   THEN3080
3090 IFR$="N"THEN3150
3100 PRINT:PRINT"■■■K■EYBOARD
   OR ■F■ILE?"
3110 GETR$:IFR$<>"K"ANDR$<>"F"
   THEN3110
3120 KF=2:IFR$="K"THENKF=1:
   GOTO3150
3130 INPUT"■INPUT FILENAME";VB$:
   VB$=LEFT$(VB$,16)
3140 IFLEFT$(VB$,1)<"A"ORLEFT$
   (VB$,1)>"Z"THEN3130
3150 PRINT"■DO YOU WISH TO CHANGE
   THE PRINTER (Y/N)?"
3160 GETR$:IFR$<>"Y"ANDR$<>"N"
   THEN3160
3170 IFR$="Y"THENGOSUB5500
3180 PRINT"■"
3190 VB=0:PP=0:AS=0:LC=1:PRINT
   "DO YOU WISH FOR A SAMPLE OUTPUT
   TO THE□□SCREEN";
3191 PRINT"□(Y/N).<RETURN> TO
   RETURN TO MAIN MENU":BL=0
3200 GETR$:IFR$<>"Y"ANDR$<>"N"
   ANDR$<>CHR$(13)THEN3200
3210 IFR$=CHR$(13)THENRETURN
3211 IFR$="N"THENP=DN:OPEN4,P,7,
   "■":GOTO3220
3215 OPEN4,3:PRINTCHR$(14)
3220 IFKF<>2THEN3240
3230 IFDL=1ANDKF=2THEN3232
3231 IFKF=2THEN3236
3232 OPEN2,8,2,VB$+",S,R":
   INPUT#2,DV,DV:GOTO3240
3236 OPEN1,1,0:INPUT#1,DV,DV
3240 GP$="":IFR$="N"THENFORF=1
   TOGP:GP$=GP$+"□":NEXT
3250 FORK=1TOTL-1
3252 IFLEFT$(TX$(K),1)="#"AND
   LEN(TX$(K))-1>AS THEN AS=
   LEN(TX$(K))
3260 NEXT:IFAS>TWTHENPRINT"ERROR:
   ADDRESS TOO LONG":GOTO3570
3270 K=1:PRINT#4,LF$;GP$;:AS$="":
   IFAS>0THENFORF=1TOGP+TW-AS:
   AS$=AS$+"□":NEXT
3280 TT$=TX$(K)
```




```
3290 IFTT$=""THENPRINT#4,CHR$(13);
    GP$;:PP=Ø:LC=LC+1:GOSUB359Ø:
    GOTO352Ø
3300 BP=Ø:FORF=1TOLEN(TT$):IF MID$
    (TT$,F,2)="]["THENBP=F:GOTO33Ø4
33Ø2 NEXT F
33Ø4 IFBP=ØORKF=ØTHEN339Ø
331Ø IFKF=1THEN337Ø
332Ø IFDL=1THEN336Ø
333Ø IFST=64THEN335Ø
334Ø INPUT#1,RP$:GOTO338Ø
335Ø PRINT"ERROR—NOT ENOUGH DATA IN
    FILE":GOTO357Ø
336Ø IFST=64THEN335Ø
3362 INPUT#2,RP$:GOTO338Ø
337Ø BL=BL+1:PRINT:RP$="":
    PRINT"INPUT VARIABLE BLOCK";
    BL;:INPUT RP$
338Ø TT$=LEFT$(TT$,BP-1)+RP$+
    MID$(TT$,BP+2):GOTO 33ØØ
339Ø CF=Ø:FORF=1TO4:IFLEFT$
    (TT$,1)=MID$("&$*#",F,1)THENCF=F
3391 NEXTF:ONCFGOTO346Ø,347Ø,349Ø,351Ø
34ØØ IFPP+LEN(TT$)<=TWTHEN
    PRINT#4,TT$;:PP=
    PP+LEN(TT$):
    GOTO352Ø
341Ø TA$=LEFT$(TT$,
    TW-PP)
342Ø CF=Ø:FORF=1TO
    LEN(TT$):IFMID$(TT$,
    F,1)="□"THEN
    CF=F:GOTO 3422
3421 NEXT F
3422 IFCF>TWTHENPRINT
    "ERROR—WORD TOO LONG IN";TT$:
    GOTO357Ø
343Ø IFRIGHT$(TA$,1)="□"THEN345Ø
344Ø IFLEN(TA$)>ØTHENTA$=LEFT$
    (TA$,LEN(TA$)-1):GOTO343Ø
345Ø PRINT#4,TA$;CHR$(13);GP$;:
    PP=Ø:LC=LC+1:GOSUB359Ø:
    TT$=MID$(TT$,LEN(TA$)+1)
3452 IFTT$<>""THENBP=1:GOTO34ØØ
3454 GOTO352Ø
346Ø PRINT#4,CHR$(13);CHR$(13);GP$;:
    PP=Ø:LC=LC+1:GOSUB359Ø:TT$=
    MID$(TT$,2):GOTO33ØØ
3465 GOTO 33ØØ
347Ø TT$=MID$(TT$,2):PRINT#4,
    CHR$(13);GP$;:IFPP<>TW
    THEN3479
3471 FORF=1TOINT(TX/2):PRINT#4,CHR$
    (32);:NEXT:PP=INT(TX/2):
    GOTO348Ø
3479 PP=Ø
348Ø LC=LC+1:GOSUB359Ø:GOTO33ØØ
349Ø TT$=MID$(TT$,2):IFLEN(TT$)
    <=TWTHEN35ØØ
3491 PRINT"ERROR—CANNOT CENTRE"
    TT$:GOTO352Ø
35ØØ PRINT#4,CHR$(13);GP$;:FOR
    F=1TOINT((TW-LEN(TT$))/2):
    PRINT#,"□";:NEXT
35Ø1 PRINT#4,TT$
35Ø6 PRINT#4,CHR$(13);GP$;:PP=Ø:
    LC=LC+1:GOSUB359Ø:GOTO352Ø
351Ø PRINT#4,CHR$(13);AS$;MID$
    (TT$,2);:PP=Ø:LC=LC+1:
    GOSUB359Ø
352Ø K=K+1:IFP=3THENFORZ=1TO
    5ØØ:NEXT
353Ø IFK<TLTHEN328Ø
354Ø IFP<>3THENPRINT#4,LF$;LF$:
    GOTO355Ø
3541 PRINT:PRINT
355Ø FORZ=1TO4:CLOSEZ:NEXTZ
```

```
3560 IFP=ØTHEN319Ø
3561 RETURN
357Ø FORZ=1TO3ØØØ:NEXT:FORZ=1
   TO8:CLOSEZ:NEXTZ
358Ø RETURN
359Ø IFLC>THTHENPRINT#4,LF$;LF$;
   GP$;:LC=1
36ØØ RETURN
```

To enter Editor mode simply press E. If you are dealing with text already stored in the memory, Press T, B or N, depending on the area of text which requires editing. If text is to be keyed in and edited at the same time, press any of these keys.

To delete characters in the work area use the right or left cursor keys to position cursor to the right of the relevant character then press DELETE.

To amend text already stored in memory, use the cursor up or down keys to move the > marker directly below the line containing the error, and press the COPY key. This will move an identical line of text into the work area. Insert or delete as described above. Return the amended line to the memory by pressing RETURN, making sure that the > marker is positioned directly below the point the amended line has to be inserted. To delete the original line, move the marker directly beneath it and press control D.

To review text in the memory, enter editor mode and press T, B or N depending on where you want to start. If you then press either the up or down cursor keys, text will move up or down ten lines at a time.

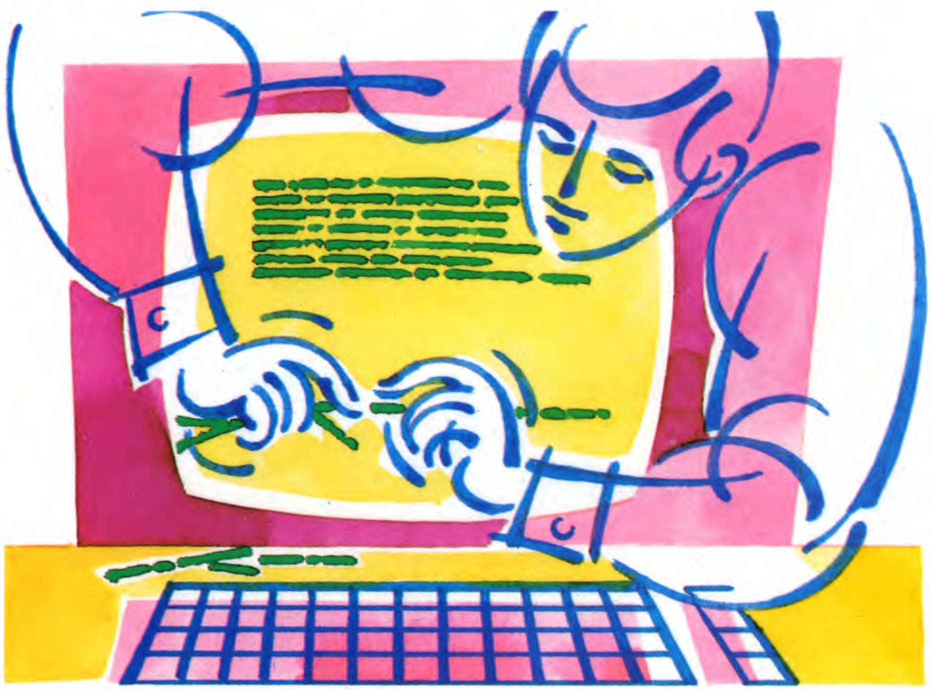
To return to the main menu press ESCAPE.

To store new text in the memory, press RETURN. Text will automatically go into memory if you key more than 12Ø characters.

Additional characters can be inserted at any point in the work area by moving the cursor to the right of the chosen point. Amendments can then be keyed in and existing text will not be overwritten.

For disk drive, omit Lines 86Ø and 1Ø2Ø.

```
67Ø TB=GET
68Ø *FX21,Ø
69Ø REM
7ØØ CP=CP+((TB=139)-(TB=138))*
   (1-9*INKEY(-1))
71Ø IF CP<1 THEN CP=1:GOTO 79Ø
72Ø IF CP>TL THEN CP=TL:GOTO 79Ø
73Ø IF TB>31 AND TB<128 OR TB=13 OR
   TB=136 OR TB=137 THEN RETURN
74Ø IF CP>1 AND TB=4 THEN TL=
   TL-1:FOR K=CP-1 TO TL:TX$(K)
   =TX$(K+1):NEXT:TX$(TL+1)=
   "":CP=CP-1:GOTO 79Ø
75Ø IF CP>1 AND TB=135 THEN VDU
   23,1,Ø;Ø;Ø;Ø;:PRINTTAB(Ø,15)
   SPC(12Ø):VDU 23,1,1;Ø;Ø;Ø;:
   A$=TX$(CP-1):RETURN
76Ø IF TB=19 THEN GOSUB 12ØØ
77Ø IF TB<>Ø THEN 79Ø
78Ø SF=SF+1:IF SF=1 THEN SS=CP
   ELSE SE=CP:SF=Ø:GOSUB 126Ø
79Ø GOSUB 6ØØ:GOTO 67Ø
8ØØ CLS:PRINTTAB(15,2)RV$"SAVE
   A FILE"NM$:IF TL=1 THEN
   PRINTTAB(Ø,1Ø)"NOTHING TO
   SAVE":FOR Z=1TO3ØØØ:NEXT:RETURN
81Ø INPUT""FILENAME PLEASE",F$
82Ø IF LEN(F$)>8 THEN PRINT"THAT NAME
   IS TOO LONG":GOTO 81Ø
83Ø CLS:PRINTTAB(Ø,4)"SAVING";
   F$
84Ø *TAPE
85Ø *OPT1,1
86Ø *OPT 2,1
87Ø IF SFF=Ø THEN *DISK
88Ø H=OPENOUT(F$)
89Ø PRINT#H,CP,TL
9ØØ FOR K=1 TO TL-1:PRINT#H,
   TX$(K):NEXT
91Ø CLOSE#H:RETURN
92Ø CLS:PRINTTAB(15,2)RV$"LOAD A
   FILE"NM$:IF TL=1 THEN 98Ø
93Ø PRINT"ARE YOU SURE (Y/N) ?"
94Ø R$=GET$:IF R$="N" THEN RETURN
95Ø IF R$<>"Y" THEN 94Ø
96Ø FOR K=1 TO TL:TX$(K)="":
   NEXT:TL=1:CP=1
97Ø TX$(Ø)=CHR$(Ø)+RV$+"START
   OF TEXT"+NM$:TX$(TL)=CHR$
   (Ø)+RV$+"END OF TEXT"+NM$
98Ø INPUT""FILENAME PLEASE",F$
99Ø IF LENF$>8 THEN PRINT"THAT NAME
   IS TOO LONG":GOTO 98Ø
1ØØØ *TAPE
1Ø1Ø *OPT 1,1
1Ø2Ø *OPT 2,1
1Ø3Ø IF LF=Ø THEN *DISK
1Ø4Ø H=OPENIN(F$)
1Ø5Ø INPUT#H,CP,TL
1Ø6Ø FOR K=1 TO TL-1:INPUT#
   H,TX$(K):NEXT
1Ø7Ø CLOSE#H
1Ø8Ø TX$(TL)=CHR$(Ø)+RV$+"END OF
   TEXT"+NM$
1Ø9Ø RETURN
11ØØ CLS:PRINTTAB(14,2)RV$"I/Ø
   SETUP"NM$
111Ø PRINT''"LOAD FROM (T)APE OR
   (D)ISK";
112Ø B$=GET$:IF B$<>"T" AND
   B$<>"D" THEN 112Ø
113Ø PRINTB$
114Ø IF B$="T" THEN LF=1 ELSE LF=Ø
115Ø PRINT''"SAVE TO (T)APE OR (D)ISK";
116Ø B$=GET$:IF B$<>"T" AND
   B$<>"D" THEN 116Ø
117Ø PRINTB$
118Ø IF B$="T" THEN SFF=1 ELSE
   SFF=Ø
119Ø RETURN
136Ø IF ERR<>17 THEN PRINT:
   REPORT:PRINT" AT LINE ";
   ERL:PRINT:END
137Ø IF ERL=21Ø THEN PRINT
   "ESCAPE":END
138Ø IF ERL<12Ø OR ERL>18Ø THEN 12Ø
   ELSE 2ØØ
```



The screen display (and eventual output) defaults to upper case (capitals). Lowercase (small) characters can be entered by releasing a cap lock using SHIFT 0. Lowercase text is shown by standard reverse field display.

The general editing controls follow the guidelines given previously. Text is edited in the work area towards the bottom of the screen—the upper part of the screen is for viewing text stored in memory. The cursor keys play an important part in moving over the text to allow editing. The left and right cursor keys enable you to move along the text line in the work area—pressing SHIFT simultaneously with left cursor 'toggles' the cursor either to the start or to the end of the line.

Further characters can be inserted at any point in the work area text by moving the cursor to the right of the chosen point. You *cannot* overwrite characters. To delete an error, position cursor over character and press down arrow.

To enter 'editor' mode press up arrow. The cursor automatically locates the last 'access' position which is indicated by a flashing > marker. The editor automatically returns to the last position of the marker.

Once in editor mode you can inspect the text in memory by scrolling upwards or downwards using up arrow and down arrow keys. Bigger (ten line) jumps are possible by pressing keys U ('up') and D ('down').

Lines of text can be deleted by positioning the marker and pressing the SHIFT and down arrow keys simultaneously. Blank lines can be inserted only in entry mode pressing ENTER with a blank line in the work area.

The line immediately above the marker can be copied to the work area (ready for editing) by pressing C in editor mode. The amended line does not automatically replace the original line when it is returned—the latter has to be deleted afterwards. You can escape from editor mode simply by pressing RETURN. This returns you to the edit mode, where CLEAR returns you to the edit mode menu.

TROUBLE SHOOTER

• With this program you can key text in a word at a time, press RETURN, and still end up with continuous text on the print out. But beware, if you don't remember to insert spaces at the end of each word, the words will be printed out as one solid line and you'll end up with gobbledegook. This is an easy mistake to make as spaces are not visible on the screen and can easily be forgotten.

```
2500 PM=5:IF CP<5 THEN PM=CP
2510 TB$=INKEY$:IF TB$="" THEN
   PRINT@PM*32,"":PRINT@PM*32,
   ">":GOTO2510
2520 CP=CP+(TB$="↑")-(TB$=
   CHR$(10))+10*((TB$="U")-
   (TB$="D"))
2530 IF CP<1 THEN CP=1
2540 IF CP>TL THEN CP=TL
2550 GOSUB2090
2560 IF TB$=CHR$(13) THEN RETURN
2570 IF CP>1 AND TB$=CHR$(91)
   THEN TL=TL-1:FORK=CP-1 TO
   TL:TX$(K)=TX$(K+1):NEXT:TX$
   (TL+1)="":CP=CP-1:GOSUB2090
2580 IF CP>1 AND TB$="C" THEN
   FORK=32 TO 1 STEP-1:IF MID$
   (TX$(CP-1),K,1)=" " THEN
   NEXT ELSE A$=LEFT$(TX$(CP-1),
   K)+" ":RETURN
2590 IF TB$="S" GOSUB5070
2600 IF TB$="@" THEN SF=SF+1:IF
   SF=1 THEN SS=CP ELSE
   SE=CP:SF=0:GOSUB5130
2610 GOTO 2500
3000 RETURN 'TEMPORARY LINE
4000 CLS:IF TL=1 THEN PRINT@7,
   "nothing to save":FORZ=1TO1000:
   NEXT:RETURN
4010 CLS:LINEINPUT" FILENAME ?";F$
4020 IF LEFT$(F$,1)<"A" OR LEFT$
   (F$,1)>"Z" THEN 4010
4030 IF TS=1 THEN 4120
4040 CLS:MOTORON:AUDIO ON:PRINT
   "POSITION TAPE, THEN PRESS enter"
4050 IF INKEY$<>CHR$(13) THEN 4050
   ELSE MOTOROFF:AUDIO OFF:PRINT
   "PLACE RECORDER IN RECORD MODE
   □□□THEN PRESS enter"
4060 IF INKEY$<>CHR$(13) THEN 4060
4070 MOTORON:FORK=1TO1000:NEXT:
   OPEN"O",#-1,F$
4080 PRINT#-1,CP,TL
4090 FOR K=1 TO TL-1:PRINT#-1,
   TX$(K):NEXT
4100 CLOSE#-1
4110 RETURN
4120 CLS:PRINT" ENSURE DRIVE IS ON AND
   A DISC□□□IS INSERTED. PRESS enter
   TO□□□□□□CONTINUE"
4130 IF INKEY$<>CHR$(13) THEN 4130
4140 CREATE F$
4150 FWRITE F$;CP:FWRITE F$;TL
4160 FOR K=1 TO TL-1
4170 FWRITE F$;TX$(K)
4180 NEXT:RETURN
4500 CLS:PRINT@8,BL$;"load";BL$;
   "a";BL$;"file";BL$:IF TL=1 THEN 4540
4510 PRINT" ARE YOU SURE (Y/N) ?"
4520 R$=INKEY$:IF R$<>"Y" AND
   R$<>"N" THEN 4520
4530 IF R$="N" THEN RETURN
4540 CLS:LINEINPUT" INPUT
   FILENAME ?";F$
4550 IF LEFT$(F$,1)<"A" OR LEFT$
   (F$,1)>"Z" THEN 4540
4560 IF DL=1 THEN 4650
4570 MOTORON:AUDIOON:PRINT
   "POSITION TAPE, PUT INTO PLAY MODE,
   THEN PRESS enter"
4580 IF INKEY$<>CHR$(13) THEN 4580
4590 OPEN"I",#-1,F$
4600 INPUT#-1,CP,TL
4610 FORK=1TOTL-1:INPUT#-1,TX$(K):
   NEXT
4620 CLOSE#-1:GOSUB2090
4630 TX$(TL)=STRING$(32,126)
4640 RETURN
4650 FREAD F$,FROM0;CP:FREAD F$;TL
4660 FORK=1 TO TL-1:FLREAD F$;TX$(K)
4670 NEXT:RETURN
5000 CLS:PRINT@10,BL$;"i";CHR$
   (124);"o";BL$;"setup";BL$:PRINT
   @96," LOAD FROM (T)APE OR
   (D)ISC ?";
5010 B$=INKEY$:IF B$<>"T" AND
   B$<>"D" THEN 5010
5020 PRINTB$:DL=0:IF B$="D" THEN
   DL=1
5030 PRINT:PRINT" SAVE TO (T)APE OR
   (D)ISC ?";
5040 B$=INKEY$:IF B$<>"T" AND
   B$<>"D" THEN 5040
5050 PRINTB$:TS=0:IF B$="D" THEN
   TS=1
5060 RETURN
5070 RETURN 'TEMPORARY LINE
5130 RETURN 'TEMPORARY LINE
5500 CLS:PRINT@8,BL$;"printer";
   BL$;"setup";BL$
5510 PRINT@128,;:INPUT" MAX. LINE
   WIDTH ";MW:MW=INT(MW):IF MW<1
   THEN 5510
5520 INPUT" LINE WIDTH REQUIRED ";
   TW:TW=INT(TW):IF TW<1 OR TW>MW
   THEN 5520
5530 INPUT" PAGE LENGTH ";PL:PL=
   INT(PL):IF PL<1 THEN 5530
5540 INPUT" TEXT LENGTH ";TH:TH=
   INT(TH):IF TH<1 OR TH>PL THEN 5530
5550 GP=INT((MW-TW)/2):LF$=STRING$
   (INT((PL-TH)/2),13)
5560 PRINT:PRINT:PRINT" IS THIS OK
   (Y/N) ?"
5570 R$=INKEY$:IF R$<>"N" AND
   R$<>"Y" THEN 5570
5580 IF R$="Y" THEN RETURN ELSE 5500
```

COMPUTER-CONTROLLED ROBOTS

Following in the wake of the Micro Revolution, the Age of Robotics has now dawned, not with the power-seeking monsters of fiction, but with helpful peripherals

Dr Frankenstein implanted a brain into a creation from the graveyard. If the mad doctor had used the brain of a computer and the body of a cyborg he would have saved himself much trouble and probably earned a Queen's Award to Industry.

Just as the human brain communicates with the outside world through the five senses for input, and through speech and movement for output—so an electronic brain needs to be linked to input and output channels. Your computer is already linked to the usual input and output devices—at least a keyboard or joystick and a monitor or TV set—but it is an ideal 'brain' for a wide range of other mechanical and electronic devices, which can provide the computer with 'senses' to receive data from the outside world.

The computer can provide 'intelligence' for a mechanical device. Connected to a typewriter (or keyboard and printer) a computer becomes a word processor; with an electronic organ, it becomes a synthesiser; linked to a vacuum cleaner, you have a robot that cleans the carpet.

Generally, computers have at least one input/output port through which they can communicate with external, mechanical devices. There are two types of port: serial and parallel, which differ according to how they handle data. In most home computers, including all those covered here, one byte of memory contains eight bits of information—hence the name '8-bit micro'. The information stored in a byte is in a string of eight Øs and 1s—for example, ØØ1Ø11Ø1.

The computer has its own internal 'memory map', in which there are two bytes of memory labelled specially for the input/output port. The first byte is the Data Direction Register (DDR). This determines the status of the port, dictating whether channels are used for input or output. On the Vic 20 for example, the address of the DDR is 37138. Its value is set using the POKE command. POKE 37138,255 sets the value to 11111111—the binary equivalent of 255. A 1 means that a channel is transmitting data; a Ø means it is receiving. So the above POKE command sets all eight channels at the user port to output. POKE 37138,15 sets the value to ØØØØ1111, which means that the first four channels are set to receive and the second four to transmit. The other computers work in a similar way, although the specific addresses to POKE are different, of course.

The second byte that controls the port is the port address. Any string of Øs and 1s in the port address can be converted by a chip on the interface board into a series of electronic voltages at the port. These voltages are then passed on to the device. At a parallel port this information passes along eight separate wires simultaneously, but at a serial port the information passes along one wire in separate bursts.

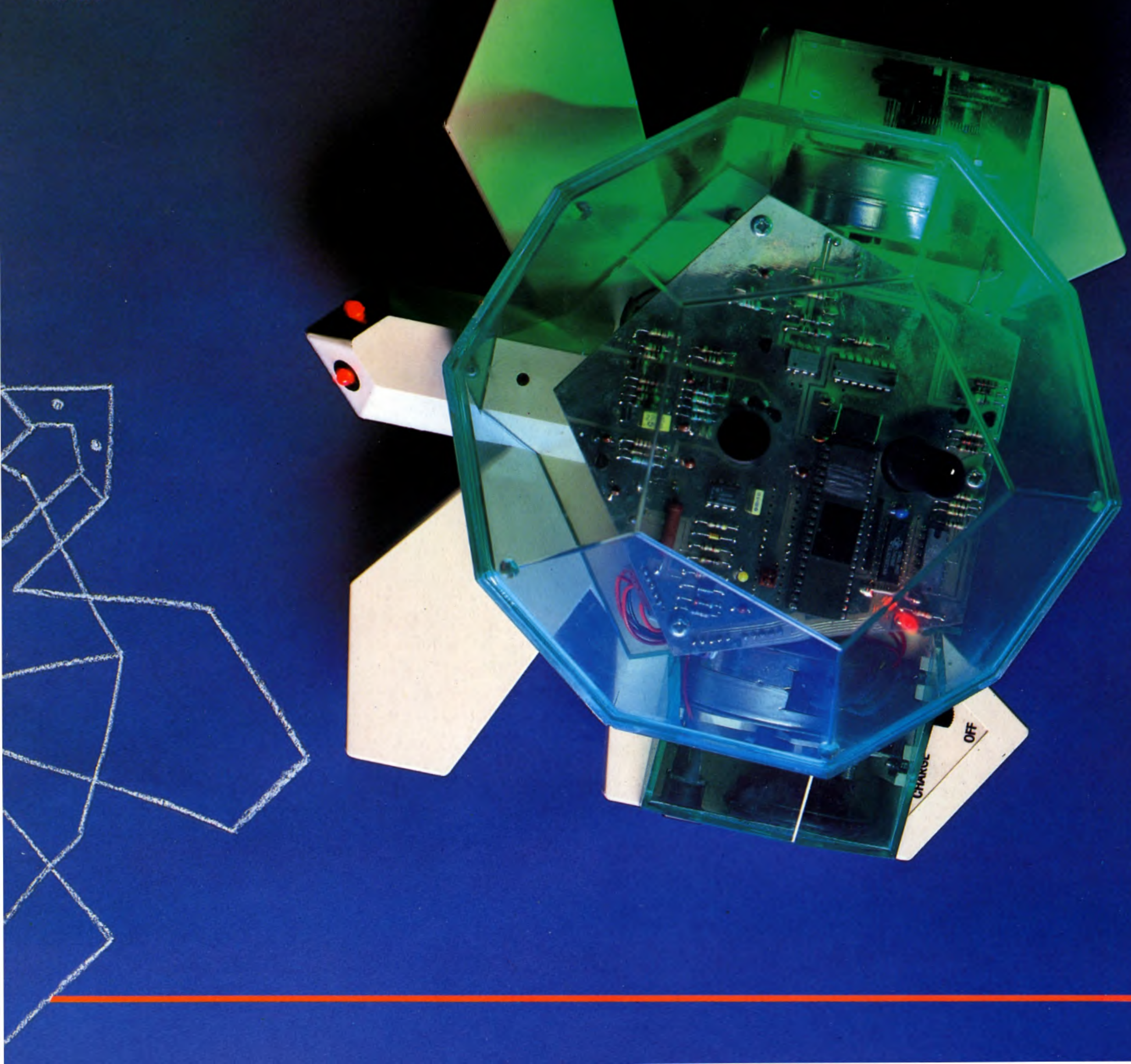
When the information reaches its destination, it is used to control various operations. Typically, in operating an electro-mechanical device such as a robot, it is used to switch different motors on and off to carry out specific functions. By setting the value of the port address, you can send signals to specific destinations. For example, if the value is set to ØØ11Ø111, a signal is sent along the 3rd, 4th, 6th, 7th and 8th wires at a parallel port, or to the 3rd, 4th, 6th, 7th and 8th destinations through a serial port.

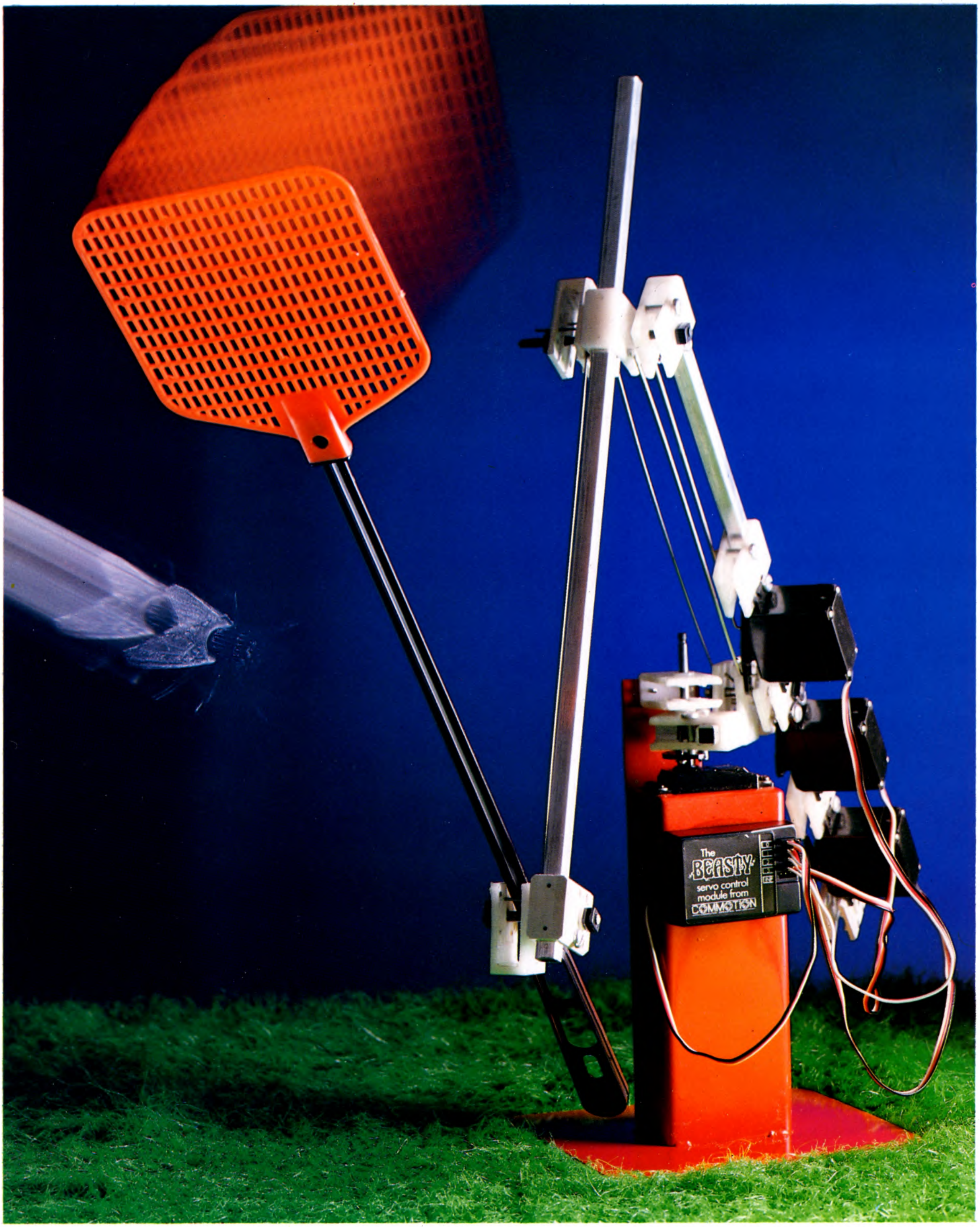


As a means of teaching the principles of programming, as well as school geometry, the Turtle and LOGO are highly successful. The remarkably good results are due in part to the affection children and adults alike have for the 'animal', which has the effect of making the subject less abstract, more understandable and, above all, a great pleasure to learn and use

TYPES OF ROBOT

Many industries use robots in situations dangerous to human beings, such as handling radioactive materials, toxic or explosive chemicals, or substances at extreme temperatures. Robots are also used in repetitive, monotonous tasks, such as paint spraying, mechanical assembly and sorting. But the use of robots is not restricted to industry. Several




The
BEASTY
servo control
module from
COMMOTION

reasonably priced robots can be controlled from a home computer. They can see, read bar codes, draw, teach a high level programming language, and manipulate objects. There are two main categories of robots suitable for home micros—robotic arms, and floor turtles and buggies. These are supplied with 'user friendly' software, requiring no specific knowledge by the user.

Robotic arms are modelled on the human arm. They can have as many as five points of movement: four of these represent the shoulder, elbow, wrist and grip. The fifth, at the arm base, gives the arm the facility to 'swivel about the hips'. Popular arms vary in price from about the price of a game on disk to about 100 times as much, so the level of sophistication is governed by how much you spend.

Armatron is one of the least expensive robotic arms. It is not programmable, so it is a sophisticated toy rather than a true robot. It is driven by a battery powered electric motor and controlled by two joysticks. At the other end of the range are Hero 1 and Genesis P101, which have their own on-board computer and are available in kit form.

In the middle of the price range (costing as much as a BBC B), Armdroid 1 is interfaced to all the computers. It is powered by six stepper motors and has five points of movement. It can raise, lower and rotate the wrist. The arm can be controlled in 'immediate' mode from the keyboard, or programmed by storing a string of arm positions in the computer memory. Movements can be run continuously or a step at a time. Pauses can be added, the arm speed altered, and routines edited. The program to move the arm is written in BASIC, with machine code calls which control the motors. The Armdroid's manufacturers provide a listing and explanation of the BASIC program, so you can modify it to suit yourself. Another electrically powered arm is the Micro Grasp, which can be interfaced to the Spectrum, Dragon and BBC.

If these are too expensive an introduction to robotic arms, try Beasty—a rather more generalized control system consisting of a small black plastic box with a row of five connectors along one edge. One of the connectors is for input, and accepts a three-core cable from a user port on your computer. As yet, Beasty is only interfaced to the BBC B, but interfaces for other home micros will soon become available.

Each of the other four connectors is for a servo unit—a high quality motor which provides feedback. The servo motors connect to Beasty with the same three-core cable that connects Beasty to the computer. In this form, Beasty is not yet a complete robot, but it has the great merit of flexibility, since the servos can be used to control almost any equipment to which they are capable of being fixed, within reach of the wires. Possible applications include all sorts of models—they have even been used for film 'monsters'—and of course robot arms. These may be home made, but Beasty's makers in fact supply a complete arm ready for attachment.

The Beasty robotic arm is powered by up to four servos. It is made with aluminium rods and plastic mouldings mounted on a metal base, and can be assembled in a number of configurations. The arm has three axes of movement plus a simple grip, and moves like a human arm with rigid wrist and hips. There is one servo for each point of movement and one for the gripper.

CONTROLLING BEASTY

Beasty's manufacturers have created a language called ROBOL to control the servos. Each program line consists of a series of instructions telling all the individual servos to make a single movement, and by how much. A complex operation can thus be broken down into a series of movements, each controlled by one program line. ROBOL is loaded from cassette and the following display appears on the screen:

```
ROBOL: Interactive robot controller
_____________________________________
1 MOVE 5ØØ   5ØØ   5ØØ   5ØØ
_____________________________________
Editing
```

This prompts you to enter the instructions which form the first program line. You do this directly from the keyboard, and the Beasty responds directly to the instructions. If you now press 1, for example, the servo plugged into position Ø on Beasty moves, and the value of the first 5ØØ on the display is increased. Pressing Q will cause the same servo to move in the opposite direction, and decrease the value of the first number. Pressing SHIFT with 1 or Q causes the values to change more quickly. Keys 2 and W, 3 and E, 4 and R, give you similar control over the movements of the remaining three servos.

Set the values of the servos, press RETURN and the screen will look something like this:

```
1 MOVE  24   344   92Ø   46Ø
2 MOVE  -    -     -     -
```

The computer has stored the values on Line 1 in its memory and is waiting for the values for Line 2. Carry on in the same way and you might produce something like this:

```
1 MOVE  24   344   92Ø   46Ø
2 MOVE  -    -     2ØØ   -
3 MOVE  -    12Ø   -     324
```

A dash indicates that a servo does not move on that particular line. In Line 2 of the above program, only the third servo moves. In Line 3, the second and fourth servos move.

There are other keywords as well as MOVE. Using JUMP makes the servo jump to its new position, a more violent action than MOVE. WAIT followed by a value causes the program to pause for a set time.

Programs can be saved on disk or tape, under the filename LIFT, for example, by pressing ESCAPE and then typing SAVE "LIFT".

Also on the tape is a machine code program called Driver which can be positioned anywhere in the computer's memory and incorporated in BASIC routines. Driver is called with the X register containing the Beasty channel number, and the Y register giving the new value for that servo. On the BBC, Driver is loaded at location &28ØØ by typing:

```
*LOAD DRIVER 28ØØ
```

The next line would be:

```
DRIVER=&28ØØ
```

which allows Driver to be called by name. So your BASIC program would begin with CALL DRIVER, to synchronize the Drivers with the computer (to initialize them).

The following program, is a typical example of how Driver can be called into a BASIC routine. But don't actually enter this, as it is only an example:

```
1Ø CALL DRIVER
2Ø REM INITIALISES DRIVER
3Ø PRINT "DO YOU WANT AN APPLE OR AN
   ORANGE?"
4Ø INPUT A$
5Ø IF A$="APPLE" THEN 1ØØ ELSE IF
   A$="ORANGE" THEN 2ØØ
1ØØ X%=Ø
11Ø REM SET SERVO TO TURN BASE
12Ø Y%=15Ø
13Ø REM TURNS BASE TO FACE APPLE
14Ø CALL DRIVER+3
15Ø REM UPDATES SERVO X% TO VALUE Y%
16Ø GOSUB 91Ø
17Ø END
2ØØ X%=Ø
21Ø Y%=Ø
22Ø REM TURNS BASE TO FACE ORANGE
23Ø CALL DRIVER+3
24Ø GOSUB 91Ø
9ØØ REM ROUTINE TO LIFT FRUIT
```

```
910 X%=1
920 REM SET SERVO TO LIFT LOWER
    SECTION OF ARM
930 Y%=200
940 REM LIFT LOWER ARM
950 CALL DRIVER+3
960 X%=2
970 REM SET SERVO TO TOP SECTION OF
    ARM
980 Y%=50
990 REM LOWER ARM TOP
1000 CALL DRIVER+3
1010 X%=3
1020 REM SET SERVO TO CLOSE GRIP
1030 Y%=260
1040 REM CLOSE GRIP
1050 CALL DRIVER+3
1060 X%=2
1070 REM SET SERVO TO TOP OF ARM
1080 Y%=255
1090 REM RAISE FRUIT
2000 CALL DRIVER+3
2010 RETURN
```

The program offers the user a choice between an apple or an orange, placed in predetermined positions in front of the Beasty arm, so the arm can move and lift the desired fruit. The positional settings depend on the way in which the arm is constructed.

OTHER FACILITIES

You can fit the Beasty arm with an electronic camera called Snap and enable your micro to 'see'. Snap contains a light sensitive chip, which transfers an image to the monitor or TV set at a resolution of 128 × 256. It weighs less than 45 g, measures 8 × 10 cm and can take up to 20 frames per second.

There are several interesting programs supplied with Snap. The program for displaying what the camera sees is called EV1. A particular frame can be kept as a still, and saved on tape or disk. And you can dump the picture on the screen to a printer. A program called Movie allows you to record and replay a series of 20 frames, creating a primitive animation sequence. One program that many users may find useful is Secure, which detects a change between a stored picture and the scene the camera is filming. This can be used as a kind of burglar alarm. When the picture has changed more than a preset amount, indicating an intruder, an alarm sounds and the program displays a graph of the number of alarms over a period of time. Another program with obvious applications in education and industry is Animal, which provides a method for the computer to recognize shapes and objects. Names are entered for different shapes. When the computer recognizes a shape, it prints out the name.

The robotic arm becomes remarkably versatile when it is mounted on a Beasty tractor base. Two servos control the direction and speed of the base engines. This only leaves two motors for the arm, but a 7 servo Beasty is under development to allow a fully operational arm to be transported on the base. Instructions travel down a spring wound cable.

THE TURTLE

One computer controlled robot that is becoming a familiar sight in primary schools is the Turtle—a cybernetic animal with a pen in its belly. The Turtle moves around the floor mimicing the graphic images on the screen. It can raise and lower its pen and be taught to draw intricate pictures and designs.

Turtles have been around for some years but have been used mostly in universities with mainframe computers. They were specifically designed to teach the programming language LOGO, which has only recently become available for microcomputers but most versions allow you to use BASIC as well. LOGO is a high level language that is particularly easy to learn. It is accessible to children as young as four years, which explains the interest from schools.

There are versions of LOGO for all the computers.

The first Turtle appeared at the Massachusetts Institute of Technology in the late 1960s, the brainchild of computer genius Seymour Papert. The first British Turtle came from Edinburgh University. It had a Meccano frame and was not accurate, but an improved version, called the Edinburgh Turtle, has since been manufactured. It is connected to the computer and power source by an umbilical cord.

Recently the Edinburgh Turtle has been upstaged by the remote-controlled Valiant Turtle. This is controlled by infra-red signals and powered by rechargable nickel cadmium batteries. Problems associated with the umbilical cord, such as it twisting and pulling the Turtle off course, are eliminated from the new remote-control version. LOGO commands are keyed into the computer where a software interface converts them into binary code. This information goes to an infra-red transmitter. The turtle picks up the signal on its infra-red receiver, and the Turtle's logic control instructs motors controlling the wheels and pen mechanism. The device actually looks like a turtle, so makes it easier for children to give directional instructions. The Turtle's eyes illuminate, serving as power indicators, dimming when the batteries need recharging. The Valiant Turtle is compatible with all versions of LOGO and is supplied with its own Turtle Graphics software so you can use BASIC if you wish. It runs on the Spectrum, ZX 81, Commodore 64, BBC B and Vic 20.

There is a second species of Turtle which inhabits the monitor. This is the screen Turtle, which sometimes looks like a tiny turtle and sometimes like a chevron. The instructions that control the floor Turtle also operate the screen turtle. Typical LOGO instructions are:

FD 200 (this moves the Turtle 200 units forward)
RT 60 (this turns the Turtle right through 60 degrees)
LT 90 (this turns the Turtle 90 degrees to the left)
PU (this raises the pen)
PD (this lowers the pen)

To draw a triangle, with sides 200 units long the instructions you type could be:

```
FD 200
RT 120
FD 200
RT 120
FD 200
RT 120
```

This could be written more economically as:

```
REPEAT 3 (FD 200 RT 120)
END
```

After defining TRIANGLE, the word becomes part of the Turtle's vocabulary. So whenever you type TRIANGLE, it will draw a triangle with sides 200 units long.

The key word TRIANGLE can be used to define a new procedure:

```
TO PATTERN
REPEAT 12 (TRIANGLE RT 30)
END
```

The keyword PATTERN could then be used in the definition of another procedure, and so on to make a complicated pattern. It is easy to see how you could define a routine to draw a leaf, for example, then another to define a branch, a tree then a forest.

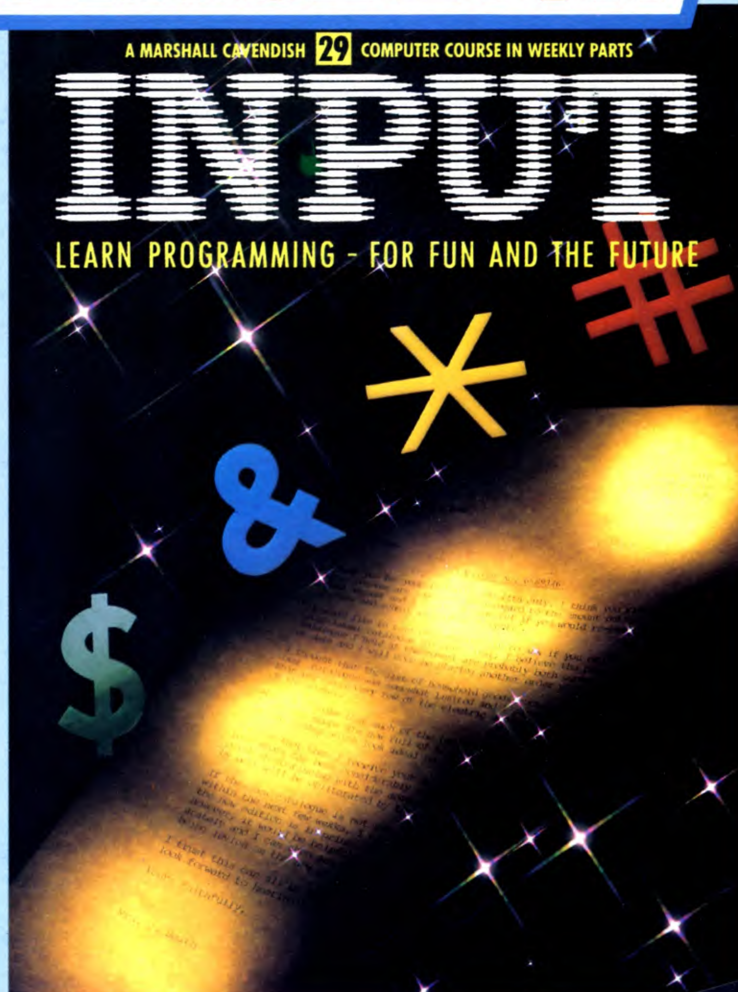
LOGO is a versatile language which can perform mathematical functions, and be used to create music. Coupled with the Turtle it makes learning computer programming fun. So the entire family can play Frankenstein, and transplant the computer's brain into mechanical devices. In doing so, they would have watched science fiction become science fact in the living room.

CUMULATIVE INDEX

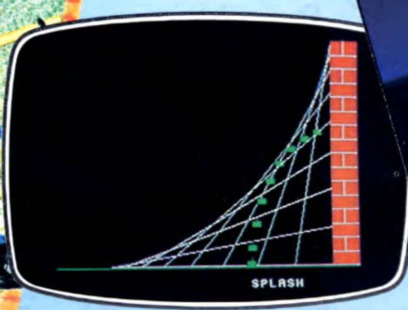
An interim index will be published each week. There will be a complete index in the last issue of *INPUT.*

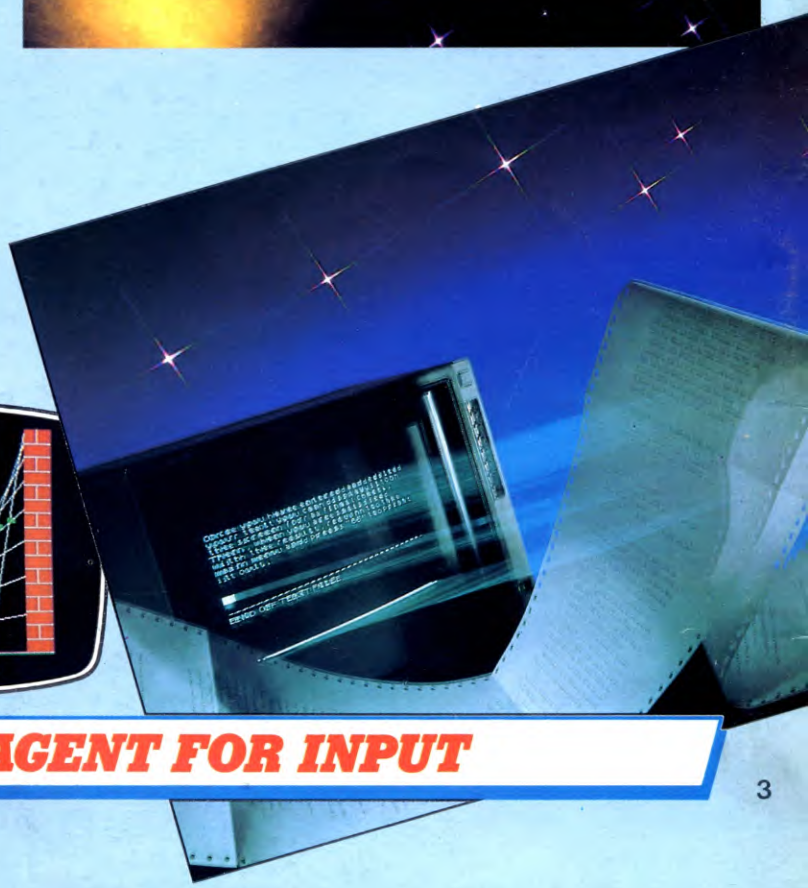
COMING IN ISSUE 29...

- *In MACHINE CODE, there's the start of CLIFFHANGER, a complete ARCADE GAME. As it builds up, you'll learn all about the routines, and create a game*
- *For intellectual types, there's a fun WORD GAME that is easy to program in BASIC and lets you play your friends at any level you want to set*
- *Continuing the mathematical background to computing, learn how to use CONIC SECTIONS IN PRACTICAL DEMONSTRATIONS*
- *There's a simple MACHINE CODE routine with instant results—it turns your TV screen into a DIGITAL CLOCK*
- *To complete the TEXT EDITOR, are SORT, SEARCH and FORM LETTER routines plus the PRINTOUT facility*









ASK YOUR NEWSAGENT FOR INPUT